Genetic Analysis of CryptoWall Ransomware

ryancor.medium.com/genetic-analysis-of-cryptowall-ransomware-843f86055c7f

Ryan Cornateanu

November 24, 2020



Nov 23, 2020

26 min read

A strain of a Crowti ransomware emerged, the variant known as CryptoWall, was spotted by researchers in early 2013. Ransomware by nature is extraordinarily destructive but this one in particular was a bit beyond that. Over the next 2 years, with over 5.25 billion files encrypted and 1 million+ systems infected, this virus has definitely made its mark in the pool of cyber weapons. Below you can find a list of the top ten infected countries:

Country	Infected systems	Percentage of total
United States	253,521	40.6%
Vietnam	66,590	10.7%
United Kingdom	40,258	6.4%
Canada	32,579	5.2%
India	22,582	3.6%
Australia	19,562	3.1%
Thailand	13,718	2.2%
France	13,005	2.1%
Germany	12,826	2.1%
Turkey	9,488	1.5%

Source: Dell Secure Works

CryptoWall is distinct in that its campaign ID initially gets sent back to their C2 servers for verification purposes. The motivation behind these ID's are to track samples by the loader vectors. The one we will be analyzing in our laboratory experiment has the crypt1 ID that was first seen around February 26th, 2014. The infection vector is still unknown today but we will be showing how to unpack the loader, and extract the main ransomware file. Some of the contagions have been caused by Drive-by downloads, Cutwail/Upatre, Infinity/Goon exploit kit, Magnitude exploit kit, Nuclear exploit kit/Pony Loader, and Gozi/Neverquest.

Initial Analysis

We will start by providing the hash of the packed loader file:

```
→ CryptoWall git:(master) openssl md5 cryptowall.binMD5(cryptowall.bin)=
47363b94cee907e2b8926c1be61150c7
```

Running the file command on the bin executable, we can confirm that this is a PE32 executable (GUI) Intel 80386, for MS Windows. Similar to the analysis we did on the Cozy Bear's Beacon Loader, we will be using IDA Pro as our flavor of disassembler tools.

Loading the packed executable into our control flow graph view, it becomes apparent fairly quickly that this is packed loader code, and the real CryptoWall code is hiding somewhere within.



WinMain CFG View

Checking the resource section of this binary only shows that it has two valid entries; the first one being a size of 91,740 bytes. Maybe we will get lucky and the hidden PE will be here?

ī	00000	6270756D	CDE 35033	12210511	CD 70 74 44	20644427	10110000	C 7 7 0 C 7 7 0			20207670	15635063	45334045	
	00000	6379756D	6D575033	43346E44	63797144	30614437	4C416366	67706E79	58546448	57645549	38397678	4F635063	4F73494E	cyummWP3C4nDcyqD0aD7LAcfgpnyXTdHWdUI89vx0cPc0sIN
	00030	77776B34	56777075	72643657	79446F50	5841642F	6A515152	4772722F	47683832	7A666B47	51503438	44303432	2F756A44	wwk4Vwpurd6WyDoPXAd/jQQRGrr/Gh82zfkGQP48D042/ujD
	00060	36665043	35504B37	616E4D58	4B6F6F51	4C6D6C4D	4F677259	75627767	507A7667	6A73484E	75516A61	73425870	39785466	6fPC5PK7anMXKooQLmlM0grYubwgPzvgjsHNuQjasBXp9xTf
	00090	5059666A	43533633	68426837	626F7979	46535143	6A534643	66434A35	6D617936	6C693935	44537441	6F724759	746E4451	PYfjCS63hBh7boyyFSQCjSFCfCJ5may6li95DStAorGYtnDQ
	00000	694B4B6D	35304D58	4D71742B	7462374C	50513650	6E6B5A63	724D6E57	772B4467	77644E78	38705473	4E4A3230	476C5368	iKKm50MXMqt+tb7LPQ6PnkZcrMnWw+DgwdNx8pTsNJ20GlSh
	000F0	46747A62	43385841	30704437	5936736D	34536B65	58437A78	32735548	50657A4B	79314148	2B6E624B	626E7049	67764659	FtzbC8XA0pD7Y6sm4SkeXCzx2sUHPezKy1AH+nbKbnpIgvFY
	00120	6367664A	4F626A75	63675864	426A4961	35766966	3863514D	69456C72	37347A78	4574776F	69423773	54614141	6B526B75	cafJObjucaXdBjIa5vif8cOMiElr74zxEtwoiB7sTaAAkRku
	00150	3774717A	34614832	47784B38	6E307642	4A4B6750	3379794D	48624840	6D58684D	38617246	4E433858	7549664F	4C625651	7taz4aH2GxK8n0vBJKaP3vvMHbHLmXhM8arFNC8XuIf0LbV0
	00180	42455773	4F733555	70334670	46636665	51536D6B	2861666E	53743277	44564(49	56376678	73613379	44574335	30644237	REWsNs511p3EpEcfe0Smk+afpSt2wDVI TV7fxsa3v1WC50dB7
	001R0	5874522R	33307260	71657859	3147534R	57646350	77583045	33754261	65464F4F	36485050	77776C2B	38593058	3728516E	XzR+30rloexY165KWicPwX0E3uBaeE0N6HPPwwl+8Y0X7+0o
	00150	496E6168	41375264	684(4371	43544557	28424846	45707177	64562B2E	40705234	6F495541	486R457A	60796051	61403768	ToohA7Ribl CoC7EW+RHEEpowdV+/IpR4pTIIAHkE71vmOal7h
	00110	66435237	ACCCC	74702548	70544154	46262651	73446230	64636542	75554657	47774662	34207067	77726860	52302570	oCP7Efp07y/Hp7A70//0sDb9dcoPullEWGwlb4.powpkmP0/p
	00210	74562064	40000E4F	78667074	46422020	97272731	F9394E47	22507720	FCCA2554	36543338	72456452	47217952	1072CE4D	
	00240	7A56396A	CAAFACER	786C707A	4043392B	382F4E74	58384F47	32307730	50042154	30343238	/34F0455	4/31/852	48726540	Zv9jjtSNxLp2FC9+8/NtX80G2PW0Vd/16128S0dSG1XRHFeM
	00270	49657346	64454058	384C414F	62665341	52325855	64773655	35374370	500/0/05	6756486B	4564416E	69783058	544A4C77	1esFdELX8LAUDTSAK2XUdw6U57CpPggegVHKEdAn1X0XIJLW
	002A0	60426778	3741-3079	54614435	47686137	6E2F752B	4A34/3/4	34697541	4848544B	6B6E6450	34623145	4A//4361	7531764E	LBgx/00y1aD5Gha/n/u+J4st41uAHKIKKndP4b1EJwCaulvN
	002D0	4D4C5231	412B6963	4F633158	6F6F526F	43636268	5143796A	784A6671	5364452F	45464773	55747277	744A2F2F	46644F34	MLR1A+icOc1XooRoCcbhQCyjxJfqSdE/EFGsUtrwtJ//FdO4
	00300	314E3341	5869687A	2F6D6E70	71503749	32304F37	4C6A6F4F	74577948	3358335A	65787551	3631626E	74676841	38494357	1N3AXihz/mnpqP7I2007LjoOtWyH3X3ZexuQ61bntghA8ICW
	00330	63554D62	71596D67	4174786D	68437570	444A4159	7A78564D	425A6D7A	6730412B	755A4B74	6E325539	494D6830	6C68797A	cUMbqYmgAtxmhCupDJAYzxVMBZmzg0A+uZKtn2U9IMh0lhyz
	00360	62353173	6E796436	37666A52	566A4247	714A594D	4F656850	737A572F	644C7349	7645425A	4D415861	5A6B6C6A	775A6B6C	b51snyd67fjRVjBGqJYMOehPszW/dLsIvEBZMAXaZkljwZkl
	00390	52765263	7A464C50	6B645159	78552F79	4B533631	676C5A73	7A2B6643	3233502F	574E785A	4B6E432B	59655939	2B717551	RvRczFLPkdQYxU/yKS61glZsz+fC23P/WNxZKnC+YeY9+quQ
	003C0	4966364A	61466A4C	354C4153	63666938	4F316E45	65774D59	34635364	4C667769	68516A6F	425A7762	66784743	554A6236	If6JaFjL5LAScfi801nEewMY4cSdLfwihQjoBZwbfxGCUJb6
	003F0	6941776C	39654830	4B685576	6A596542	6F7A6632	47343670	316A4263	494B6A38	6C6D7444	78624849	4B78384F	72456652	iAwl9eH0KhUvjYeBozf2G46p1jBcIKj8lmtDxbHIKx80rEfR
	00420	4E4C6838	377A2F6F	4C634862	66326863	2F627256	5244656E	79475065	50635658	46465749	5274356B	74714333	36646675	NLh87z/oLcHbf2hc/brVRDenvGPePcVXFFWIRt5ktgC36dfu
	00450	5554652B	59674F31	306D3137	6E4F4D6B	33306542	57543243	6758434C	34584A4E	62754345	3731726F	75667746	35485361	UTe+Ya010m17n0Mk30eBWT2CaXCL4XJNbuCE71roufwF5HSa
	00480	63456366	626F3055	37765271	69414364	41495153	2B786B68	435A306E	62465232	6A706178	4538687A	6C4A2F34	6D694370	cEcfbo0U7vRaiACdAIOS+xkhCZ0nbFR2ipaxE8hzl]/4miCp
	004R0	59344778	6F797441	45757446	4779524R	72445151	4C612E44	70746E33	654D354F	7344524F	4D416E65	4F742F58	74624570	Y4GxovtAEuzEGvRKr100Lg/Dpzp3eM5NsDRNMApe0z/XthEp
	00450	53736(67	77456171	68646674	79564660	40784072	562F7077	28735230	56376833	584F4270	48605333	56656F44	2F4C3845	SslawEaghdftyVEmLxLrV/nw+sR0V7k3X0RnHmS3VeoD/L8E
	00510	33754037	62307450	36485547	54652B54	5032332E	6276634D	74736968	31774157	65474863	43584566	67775755	52497152	3uM7b0zD6HIIG7e+TD23/bucMtsib1wAWeCHcCYEfawWIIDIaD
	00510	64424170	21455771	35786074	60504926	A66E2D72	49244072	22605168	46447442	60652748	60715926	49794553	72427446	jPAn1NWaExkzkYU/EntcHAMn2j0hElzCin7UiaY6Ux0ScPzE
	00540	40407771	20204627	4DC2E8E0	62597967	400E2B73	46344072	76546460	404A7A45	42457045	E2744CCE	77225674	74727670	JBAPINIGSXK2KTH/FH+Sh4MF2LQHFJ2CLH/HLQXORXOSSB2F
	00570	4040///1	20204037	40023630	03307007	7173333A	41443402	70340A09	31030U4E	4343704E	34313550	72672065	26496045	VUI12 /histrue obsols /wttc/rso1w2pes/t415/so0s(Ui0
	DACOD	50555152	2FOBOA72	5460482B	6F4F6270	5040782F	5/54494/	506055559	51575A52	03734849	54515559	7207390E	5048094F	VUI2/KJrimH+OUDP0FX/WIIGVmS91W2RCSK1415frg9h6h10
	00500	567A7354	55494759	38764E65	66374868	42496730	78///A/A	395A4772	632F6773	5949716C	46423031	60503475	445A5367	VZSTUIGY8VNet/KhBig0xwZZ9ZGrc/gstiqLFB01mP4uDZSg
	00600	5A345754	6C79362B	3850444A	39536838	6F496B45	4D5A3748	717A304C	624C4B76	74446852	49413153	44554662	75624164	Z4WILy6+8PDJ9Sn801KEMZ/HqZ0LDLKVtDnR1A1SDUFDuDAd
1	00630	50707455	78457256	69637639	76555671	40786276	4856336B	78744E34	57357474	28262069	496D6C62	57653231	56714F49	PptUxErVicv9vUVqMxbvKV3kxtN4W5tt+VPiImlbWe21Vq01
	00660	71756342	59446E44	33546B4B	61664673	4D314A4E	63767351	33727956	74765161	66485867	396B6B54	64776D52	534A3770	qucBYDnD3TkKafFsM1JNcvsQ3ryVtvQafHXg9kkTdwmRSJ7p
	00690	67415164	36646D75	2F772B6E	68574933	476D5577	6B426156	5669674A	474B4430	6A304E35	63716E41	49777547	7039544A	gAQd6dmu/w+nhWI3GmUwkBaVVigJGKD0j0N5cqnAIwuGp9TJ
	006C0	4C4C4172	357A5556	4A626E54	6C6F4255	67724C4E	50364F66	77506657	4E622B6F	44463263	50736263	75336C78	6864396D	LLAr5zUVJbnTloBUgrLNP60fwPfWNb+oDF2cPsbcu3lxhd9m
	006F0	53545773	30686676	55745A6B	46567232	31303833	4B535934	46576944	32647846	5A714F36	35314D32	73665049	33477953	STWs0hfvUtZkFVr21083KSY4FWiD2dxFZq0651M2sfPI3GyS
	00720	57553658	4D58574D	35756869	67364F56	3848774A	7978446B	41595766	636B7777	555A3470	43334272	54532B69	304F3743	WU6XMXWM5uhig60V8HwJyxDkAYWfckwwUZ4pC3BrTS+i007C
	00750	686C727A	4D496739	324C4F6B	58474C5A	45663448	4F436744	4A562F4B	716A434B	63354A78	737A7451	4A667A56	307A5261	hlrzMIg92L0kXGLZEf4H0CgDJV/KqjCKc5JxsztQJfzV0zRa
	00780	764B5967	50572F54	742B6E46	694E344E	77455264	51526E75	382B2F6E	78624E33	576F6C77	2B466237	6C576774	574F4530	vKYgPW/Tt+nFiN4NwERdQRnu8+/nxbN3Wolw+Fb7lWgtW0E0
	007B0	7A344B35	6C756F41	592F5847	33426B4A	62504274	4C6E5633	5A666F54	30363855	70764D4E	66316266	41545557	64667957	z4K5luoAY/XG3BkJbPBtLnV3ZfoT068UpvMNf1bfATUWdfyW
	007E0	5433335A	66776363	774E6978	7778416D	69517A73	47614D76	6B785747	564A7444	48343933	67575A34	6F4B6A44	49687546	T33ZfwccwNixwxAmiQzsGaMvkxWGVJtDH493aWZ4oKiDIhuF
	00810	707A3536	4C354E37	366A5254	4A4B7A53	716F3758	764C584D	4C794F69	77463556	4F4C7843	432F4676	51635866	67327877	pz56L5N76jRTJKzSqo7XvLXMLyOiwF5V0LxCC/Fv0cXfq2xw
	00840	314E3570	53447572	74477279	55646C62	47466D39	306D4B32	53795537	61567543	334D7849	5A667135	316E474D	6539634F	1N5pSDurzGrvUdlbGFm90mK2SvU7aVuC3MxIZfa51nGMe9c0
	00870	34307546	52516257	36366150	5244326B	512F444F	75516B61	36326964	41622F66	69776D4B	4370486F	502F3678	64315778	40uFRObW66aPRD2k0/DNuOka62ijAb/fiwmKCpKoP/6xj1Wx
1	00010	10110000	20524570	12121000	CDCCCCCC	1071 1000	CE707444	12251116	71705110	72626050	CE 4 C 74 40	CEDCEDEE	76724655	

Dumped resource section

Unfortunately not! This looks like some custom base64 encoded data that will hopefully get used later somewhere down the line in our dissection of the virus. If we scroll down to the end of WinMain() you'll notice a jump instruction that points to EAX. It will look something like this in the decompiler view:

```
JUMPOUT(eax=decrypted_code_segment);
```

Unpacking Binary Loaders

At this point, we have to open up a debugger, and view this area of code as it is being resolved dynamically. What you will want to do is a set a breakpoint at 0x00402dda, which is the location of the jmp instruction. Once you hit this breakpoint after continuing execution, you'll notice EAX now points to a new segment of code. Dumping EAX in the disassembler will lead you to the 2nd stage loader. Use the debugger's step into feature, and our instruction pointer should be safely inside the decrypted loader area.



2nd Stage

Let's go over what is happening at this stage of the malware. EBP+var_EA6E gets loaded effectively into EDX , EAX then holds the index count incrementer to follow the next few bytes at data address 302C9AEh .

.data:0302CA46 mov bl, byte ptr (loc_302C9AE - 302C9AEh)[eax].data:0302CA48 add ebx, esi.data:0302CA4A mov [edx], bl

All this snippet of code is doing is loading bytes from the address mentioned above and storing it at **b1** (the lower 8 bits of **EBX**). The byte from **b1** is then moved into the pointer value of **EDX**. At the end of this routine **EBP+var_EA6E** will hold a valid address that gets called as **EAX** (we can see the line highlighted in red in the image above). Stepping into **EAX** will now bring us to the third stage of the loading process.

A lot is going on at this point; this function has a couple thousand lines of assembly to go over, so at this point it's better we open the decompiler view to see what is happening. After resolving some of the strings on the stack, there is some key information that starts to pop up on the resource section we viewed earlier.

```
pLockRsrc = GetProcAddress(kernel32, &LockResource);pSizeofResource =
GetProcAddress(kernel32, &SizeofResource);pLoadResource = GetProcAddress(kernel32,
&LoadResource);pGetModuleHandle = GetProcAddress(kernel32,
&GetModuleHandleA);pFindRsrc = GetProcAddress(kernel32, &FindResourceA);pVirtualAlloc
= GetProcAddress(kernel32, &VirtualAlloc);
```

The malware is loading all functions dynamically that have to do with our resource section. After the data gets loaded into memory, CryptoWall begins its custom base64 decoding technique and then continues to a decryption method as seen below.







Most of what is happening here can be explained in a <u>decryptor</u> I wrote that resolves the shellcode from the resource section. If you head over to the python script, you'll notice the custom base64 decoder is fairly simple. It will use a hardcoded charset, and check to see if any of the bytes from the resource section match a byte from the charset; if it is a match, it breaks from the loop. The next character gets subtracted by one and compared to a value of zero, if greater, it will take that value and modulate by 256 ; that byte will then get stored in a buffer array. It will perform this in a loop 89,268 times, as that is the size of the encoded string inside the resource section.

Secondary to this, another decryption process starts on our recently decoded data from the algorithm above. Looking at the python script again, we can see that <u>hardcoded</u> <u>XOR</u> <u>keys</u> were extracted in the debugger if you set a breakpoint inside the decryption loop. All that is happening here is each byte is getting decrypted by a rotating three byte key. Once the loop is finished, the code will return the address of the decrypted contents, which essentially just contains an address to another subroutine:

```
loop: buffer = *(base_addr + idx) - (*n ^ (&addr + 0xFFE6DF5F + idx));
(base_addr + idx++) = buffer;...Fourth_Stage_Loader = base_addr;return
(&Fourth_Stage_Loader)(buffer, b64_decoded_str, a1);
```

The base_addr transfers data to another variable that we named Fourth_Stage_Loader which holds the address of the newest function, and can be used as a caller. If we dump the address at call dword ptr gs:(loc_1920A1-1920A1h)[eax] into memory, you'll see bytes that start with a generic x86 function prologue like 55 8b ec 81. Dump this to a file, and we can actually emulate this <u>shellcode</u>. In doing so, we don't have to step through all this code in the debugger; instead it will hopefully tell us how to unpack and get to the main CryptoWall file.

Side note: the python script I wrote will automatically decode & decrypt the resource section, and dump it to a bin file by running => <u>python decrypt shellcode loader.py -e</u>.

```
0x1000: push ebp0x1001: mov ebp, esp0x1003: add esp, 0xfffff004....
```

An easy way to see what this next stage in the malware's loader is doing is by using one of my favorite shellcode emulator tools called <u>ScDbg</u>. By using this tool, we can figure out exactly where we need to set our breakpoints in order to get to the main ransomware file. We are going to look for calls such as <u>VirtualAlloc</u>, <u>WriteProcessMemory</u>, <u>CreateProcessA</u>, etc.

C:\> scdbg.exe /s 3200000 /bp WriteProcessMemory /f dump.binLoaded 10587 bytes from file extractions/pe_process_injector_dump.binBreakpoint 0 set at 7c802213Initialization Complete..Max Steps: 3200000Using base offset: 0x4010004011cf GetProcAddress(LoadLibraryA)40165f GetProcAddress(VirtualAlloc)401c46 GetProcAddress(GetCurrentProcessId)401c52 GetCurrentProcessId() = 29...401d46 CloseHandle(18be)401f40 VirtualAlloc(base=0 , sz=20400) = 6000004021e1 VirtualAllocEx(pid=1269, base=400000 , sz=25000) = 621000 /* Breakpoint 0 hit at: 7c802213 */ 4021fe WriteProcessMemory(pid=1269, base=400000 , buf=600000, sz=400, written=12fd70) /* Breakpoint 0 hit at: 7c802213 */40224e WriteProcessMemory(pid=1269, base=401000 , buf=600400, sz=16400, written=12fd70)

Interesting... it looks like the malware is allocating memory to its own process by using GetCurrentProcessId() and allocating a large enough space to inject a PE file into itself. After memory allocation, CryptoWall injects the payload file twice, once for the header, and the second time for the rest of the file. If you set a breakpoint at WriteProcessMemory, and continue execution twice, you can dump the second argument (ECX) on the stack to see the hidden PE file.



There is an Anti-VM trick along the way in the 3rd stage part of the loader process that needs to be patched in order to hit the injection process, so I wrote an <u>x32Dbg python plugin</u> to help automate the patching and dumping operation.

Reversing the Main Crypto Binary

CryptoWall's entry point starts off by dynamically resolving all imports to obtain all of NTDLL's offsets by using the process environment block.

	- 1 iı	nt ResolveImportsFromDLL()
	2 {	
		<pre>int v1; // [esp+0h] [ebp-10h]</pre>
		int isSuccess; // [esp+4h] [ebp-Ch]
		<pre>int (stdcall *NtAllocateVirtualMemory)(signed int, int *, _DWORD, int *, signed int, signed int); // [esp+8h] [ebp-8h]</pre>
		int *ntdll_base_addr; // [esp+Ch] [ebp-4h]
		isSuccess = 0;
		<pre>ntdlL_base_addr = FindOffsetOfDLl(0x84C05E40);</pre>
•		if (ntall_base_addr)
		t Constinue at a final field of the field of
		NCALLOCATEVITUALMEMORY = SearChullForrunctions(ntall_base_dat, %xu6z0A5/4);
•		r (NEALLOCATEVITEUALMEmory)
•		v1 = 1064 ·
		if (NtAllocateVirtualMemory(-1 & function ptr 0 &v1 12288 4))// NtAllocateVirtualMemory
	18	
•	19	*function ptr = SearchDLLForFunctions(ntdl) base addr. 0x180(0D23):// NtClose
•		*(function ptr + 4) = SearchDLLForFunctions(ntdl) base addr. 0x183679F2):// ldrLoadDLL
•		*(function_ptr + 8) = SearchDLLForFunctions(ntdll_base_addr, 0x864C13EE);// ldrGetProcAddress
•		<pre>*(function_ptr + 12) = NtAllocateVirtualMemory;</pre>
•		*(function_ptr + 16) = SearchDLLForFunctions(<u>itdll_base_addr</u> , 0xF97A25D4);// ZwFreeVirtualMemory
٠		*(function_ptr + 20) = SearchDLLForFunctions(ntdll_base_addr, 0xD2654135);// ZwProtectVirtualMemory
٠		*(function_ptr + 24) = SearchDLLForFunctions(ntdll_base_addr, 0xE8B3559);// NtQueryVirtualMemory
٠		*(function_ptr + 28) = SearchDLLForFunctions(ntdll_base_addr, 0xE9FA5FEC);// NtWriteVirtualMemory
٠		<pre>*(function_ptr + 32) = SearchDLLForFunctions(ntdll_base_addr, 0x918ED998);// NtReadVirtualMemory</pre>

It will then call a subroutine that is responsible for using the base address of the loaded DLL and uses many hardcoded DWORD addresses to locate hundreds of functions.

Side Note: If you would like to make your life a whole lot easier with resolving the function names in each subroutine, I made a local type definition for IDA Pro over <u>here</u>. The resolving import function table will look a lot cleaner than what you see above:



After the function returns, the malware will proceed to generate a unique hash based on your system information, the resulting string will be MD5 hashed => DESKTOP-QR18J6QB0CBF8E8Intel64 Family 6 Model 70 Stepping 1, GenuineIntel . After computing the hash, it will setup a handle to an existing named event object with the

specified desired access that will be called as

\\BaseNamedObjects\\C6B359277232C8E248AFD89C98E96D65 .

The main engine of the code starts a few routines after the malware checks for system information, events, anti-vm, and running processes.

Most of the time the ransomware will successfully inject its main thread into svchost and not explorer ; so let's follow that trail. Since this is a 32-bit binary its going to attempt to find svchost.exe inside of SysW0W64 instead of System32. After successfully locating the full path, it will create a new thread using the RtlCreateUserThread() API call. Once the thread is created, NtResumeThread() will be used on the process to start the ransomware_thread code. Debugging these types of threads can be a little convoluted, and setting breakpoints doesn't always work.

.text:00416F40 ransomware_thread proc near .text:00416F40 start+86↓o.text:00416F40.text:00416F40 var_14 = dword ptr -14h.text:00416F40 = dword ptr -10h.text:00416F40 var 10 var C = dword ptr -0Ch.text:00416F40 var_8 = dword ptr -8.text:00416F40 = dword ptr -4.text:00416F40.text:00416F40 000 var_4 push ebp.text:00416F41 004 mov ebp, esp.text:00416F43 004 sub esp, 14h.text:00416F46 018 call ResolveImportsFromDLL...

Using x32Dbg, you can set the EIP to address 0x00416F40 since this thread is not resource dependent on any of the other code that has been executed up until this point; this thread even utilizes the ResolveImportsFromDLL function we saw in the beginning of the program's entry point... meaning, the forced instruction pointer jump will not damage the integrity of the ransomware.

```
isHandleSet = SetSecurityHandle();if ( isHandleSet && SetupC2String() ){ v8 = 0;
v6 = 0; IsSuccess = WhichProcessToInject(&v8, &v6); if ( IsSuccess ) {
IsSuccess = StartThreadFromProcess(-1, InjectedThread,
0, 0, 0); FreeVirtualMemory(v8); }}
```

The thread will go through a series of configurations that involve setting up security attributes, MD5 hashing the hostname of the infected system, and then searching to either inject new code into svchost or explorer. In order to start a new thread, the function WhichProcessToInject will query the registry path, and check permissions on what key values the malware has access to. Once chosen, the InjectedThread process will resume. Stepping into that thread, we can see the module size is fairly small.

.text:00412E80 InjectedThread proc near ; DATA .text:00412E80 .text:00412E80 000 push ebp.text:00412E81 004 mov ebp, esp.text:00412E83 004 call MainInjectedThread.text:00412E88 004 push 0.text:00412E8A 008 call ReturnFunctionName.text:00412E8F 008 mov eax, [eax+0A4h].text:00412E95 008 call eax.text:00412E97 004 eax, eax.text:00412E99 004 ebp.text:00412E9A 000 xor pop InjectedThread endp retn.text:00412E9A

At address $0 \times 00412E83$, a subroutine gets called that will bring the malware to start the next series of functions that involves the C2 server configuration callback, and the encryption of files. After the thread is finished executing, EAX resolves a function at offset $+0 \times 0A4$

which will show RtlExitUserThread being invoked. Once we enter MainInjectedThread, you'll notice the first function at 0x004011B40 is giving us the first clue of how the files will be encrypted.

.text:00	411D06	06C		push	0F0000000h.text:00411D0B 070				
push	1.text:	00411D0D	074		lea	edx,			
[ebp+reg	_crypt_	_path].tex	t:00411D10	074		pu	ısh	edx.text:00411D11 (078
push	0.text:	00411D13	07C		lea	eax,	[ebp+	-var_8].text:00411D10	6
07C		push	eax.te>	kt:004110	017 080			call	
ReturnFu	nctionN	lame.text:	00411D1C 08	30		mov	e	ecx,	
[eax+240	h].text	:00411D22	080		call	ecx	; Cry	/ptAcquireContext	

CryptAcquireContext is used to acquire a handle to a particular key container within a particular cryptographic service provider (CSP). In our case, the CSP being used is Microsoft\Enhanced\Cryptographic\Provider\V1, which coincides with algorithms such as DES, HMAC, MD5, and RSA.

```
ptContext = 0;
if ( KeyFromCryptographicServiceProvider(&cryptContext) )
  OriginalKey = AllocateSetMemory(16);
  if ( OriginalKey )
   rsa_key = 0;
   rsa_key_size = 0;
   AddComputerHashToRegistrySoftwarePath();
   isRequestSuccess = WhichProcessToInject(&rsa_key, &rsa_key_size);
   v37 = CreateTextForRansomwareNote(0, 0, 0);
   if ( !isRequestSuccess || !v37 )
      remaining_c2_data = 0;
     while (1)
       isRequestSuccess = SecondRequestToC2(&rsa_key, &rsa_key_size, &remaining_c2_data);
       if ( isRequestSuccess )
         break;
       sleep(0x1388u);
```

Once the CryptoContext is populated, the ransomware will use the MD5 hash created to label the victim's system information and register it as a key path as such → software\\C6B359277232C8E248AFD89C98E96D65. The ransom note is processed by a few steps. The first step is to generate the TOR addresses which end up resolving four addresses: http[:]//torforall[.]com, http[:]//torman2[.]com, http[:]//tormana[.]com, and http[:]//torroadsters[.]com. These DNS records will be used later on to inject into the ransomware HTML file. Next, the note gets produced by the use of the Win32 API function, RtlDecompressBuffer, to decompress the data using COMPRESSION_FORMAT_LZNT1. The compressed ransom note can be found in the .data section and consists of 0x52B8 bytes.

.data:0041AF90	compressed_ransome	Note di	0B4h	
.data:0041AF91	db	52h		
.data:0041AF92	db	0		
.data:0041AF93	db	0		
.data:0041AF94	db	25h		
.data:0041AF95	db	0B5h	; µ	
.data:0041AF96	db	0		
.data:0041AF97	db	OFDh ;	;Ý	
.data:0041AF98	db	9Fh	; Ÿ	
.data:0041AF99	db	60h		
.data:0041AF9A	db	OFDh ;	;Ý	
.data:0041AF9B	db	76h		
.data:0041AF9C	db	20h		
.data:0041AF9D	db	0		
.data:0041AF9E	db	0		
.data:0041AF9F	db	0		
.data:0041AFA0	db	OFFh ;	ÿΥ	
.data:0041AFA1	db	OFEh	; þ	
.data:0041AFA2	db	3Ch	; <	
.data:0041AFA3	db	0		
.data:0041AFA4	db	68h	; h	
.data:0041AFA5	db	0		
.data:0041AFA6	db	74h	; t	
.data:0041AFA7	db	0		
.data:0041AFA8	db	0		
.data:0041AFA9	db	6Dh		
.data:0041AFAA	db	0		
.data:0041AFAB	db	6Ch	; 1	
.data:0041AFAC	db	0		
.data:0041AFAD	db	3Eh	; >	
.data:0041AFAE	db	0		
.data:0041AFAF	db	ODh		
.data:0041AFB0	db	0		

Decompressing the note is kind of a mess in python as there is no built in function that is able to do LZNT1 decompression. You can find the actual call at address 0x004087F3.

```
.text:004087CF 024
                                   lea
                                           ecx, [ebp+var_8].text:004087D2 024
push
        ecx.text:004087D3 028
                                              mov
                                                      edx, [ebp+arg_4].text:004087D6
028
                    push
                            edx.text:004087D7 02C
                                                                  mov
                                                                           eax,
[ebp+arg_6].text:004087DA 02C
                                              push
                                                      eax.text:004087DB 030
mov
        ecx, [ebp+var_18].text:004087DE 030
                                                                    ecx.text:004087DF
                                                            push
034
                            edx, [ebp+var_C].text:004087E2 034
                    mov
                                                                                push
edx.text:004087E3 038
                                              eax, [ebp+var_12].text:004087E7 038
                                      movzx
        eax.text:004087E8 03C
push
                                              call
ReturnFunctionName.text:004087ED 03C
                                                     mov
                                                             ecx,
[eax+178h].text:004087F3 03C
                                                     ecx// Decompiled below(*
                                             call
(RtlDecompressBuffer))(COMPRESSION_FORMAT_LZNT1,
uncompressed_buffer,
                                              UncompressedBufferSize,
CompressedBuffer,
                                           CompressedBufferSize,
FinalUncompressedSize) )
```

After the function call, <u>uncompressed_buffer</u> will be a data filled pointer to a callerallocated buffer (allocated from a paged or non-paged pool) that receives the decompressed data from *CompressedBuffer*. This parameter is required and cannot be **NULL**, which is why there is an NtAllocateVirtualMemory() call to this parameter before being passed to decompression. The <u>script</u> I wrote will grab the compressed data from the PE file, and run a <u>LZNT1 decompression algorithm</u> then place the buffer in an HTML file. The resulting note will appear on the victims system as such:

A or join this there produces by a strong encryption hair reserves casing on production above. Agree information above the encryption keys using RSA-2048 can be found here: <u>http://en.wkipedia.org/wki(RSA_(cryptosystem)</u> What does this mean? This means that the structure and data within your files have been irrevocably changed, you will not be able to work with them, read them or see them, it is the same thing as losing them forever, but with our help, you can restore them. How did this happen? Expecially for you, on our server was generated the secret key pair RSA-2048 - public and private. If your files were encrypted with the public key, which has been transferred to your computer via the Internet. Secrypting of your files is only possible with the help of the private key and decrypt program, which is on our secret server. What do ldo? Uas, if you do not take the necessary measures for the specified time then the conditions for obtaining the private key will be changed. I you really value your data, then we suggest you do not waste valuable time searching for other solutions because they do not exist. For more specific instructions, please visit your personal home page, there are a few different addresses pointing to your page below: 1.%SERWICE_WEB_1% 2.%SERWICE_WEB_2% 3.%SERWICE_WEB_2% 3.%SERWICE_WEB_4% If for some reasons the addresses are not available, follow these steps: 1. Download and install tor-browser: <u>http://www.topproject.org/projects/hotbrowser.html.en</u> 2. After a successful installation, run the browser and walt for initialization. 3. Type in the address are not available, follow these steps: 4. Follow the instructions on the site. MPORTIANT INFORMATION: Your Personal PAGE <u>%SERVICE_TOR%</u> Your Personal PAGE <u>%SERVICE_WEB_1%</u> Your Personal PAGE <u>%SERVICE_TOR%</u>	I of your file	new or your instantial by a stream ascention with DSA-20.58 using CounterWall 3.0
What does this mean? This means that the structure and data within your files have been irrevocably changed, you will not be able to work with them, read them or see them, it is the same thing as losing them forever, but with our help, you can restore them. tow did this happen? Specially for you, on our server was generated the secret key pair RSA-2048 - public and private. dy our files were encrypted with the public key, which has been transforred to your computer via the internet. Decrypting of your files is only possible with the help of the private key and decrypt program, which is on our secret server. What do tho? Nas. if you do not take the necessary measures for the specified time then the conditions for obtaining the private key will be changed. I you really value your data, then we suggest you do not waste valuable time searching for other solutions because they do not exist. For more specific instructions, please visit your personal home page, there are a few different addresses pointing to your page below: 1.% SERVICE_WEB_1% 2.% SERVICE_WEB_2% 3.% SERVICE_WEB_3% If for some reasons the addresses are not available, follow these steps: 1. Openhoad and install tor-browser: http://www.topproject.org/projects/torbrowser.html.en 2. Abe a successful installation, run the browser and walt for initialization: 3. Type in the addresses are not available, follow these steps: 4. Follow the instructions on the site	Acre informu	to were protected by a serving encryption with PGP-2040 using Gryptoman 3.0. International services and the service of the ser
What does this mean? This means that the structure and data within your files have been irrevocably changed, you will not be able to work with them, read them or see them, it is the same thing as losing them forever, but with our help, you can restore them. How did this happen? Expecially for you, on our server was generated the secret key pair RSA-2048 - public and private. Wow did this happen? Expecially for you, on our server was generated the secret key pair RSA-2048 - public and private. Was of your files is only possible with the help of the private key and decrypt program, which is on our secret server. What do 1 do? Was, if you do not take the necessary measures for the specified time then the conditions for obtaining the private key will be changed. For more specific instructions, please visit your personal home page, there are a few different addresses pointing to your page below: 1.%SERVICE_WEB_1% 2.%SERVICE_WEB_2% 3.%SERVICE_WEB_3% 4.%service_weB_4% If for some reasons the addresses are not available, follow these steps: 1. Norenload and install tor-browser: http://www.teproject.org/projects/lorbrowser.html.en 2. After a successful installation, run the browser and wait for initialization. 3. Type in the addresses if %SERVICE_TOR% 4. Follow the instructions on the site. <td< td=""><td></td><td>and about the one point reporting new concerter to the transmission in the intervention of the point of the p</td></td<>		and about the one point reporting new concerter to the transmission in the intervention of the point of the p
his means that the structure and data within your files have been irrevocably changed, you will not be able to work with them, read them or see them, it is the same thing as losing them forever, but with our help, you can restore them. tow did this happen? Especially for you, on our server was generated the secret key pair RSA-2048 - public and private. We will not a server was generated the secret key pair RSA-2048 - public and private. We will not a secret server. What do I do? Was, if you do not take the necessary measures for the specified time then the conditions for obtaining the private key will be changed. I you really value your data, then we suggest you do not waste valuable time searching for other solutions because they do not exist. For more specific instructions, please visit your personal home page, there are a few different addresses pointing to your page below: 1.%SERVICE_WEB_1% 2.%SERVICE_WEB_2% 3.%SERVICE_WEB_2% 3.%SERVICE_WEB_3% 4.%SERVICE_WEB_4% If or some reasons the addresses are not available, follow these steps: 1. Download and install tor-browser: <u>http://www.torproject.org/projects/hotprovser.html.en</u> 2. After a successful installation, run the browser and wait for initialization. 3. Type in the address bar: %SERVICE_TOR% 4. Follow the instructions on the site. MPORTANT INFORMATION: Your Personal PAGE: <u>%SERVICE_TOR%</u>	What does t	his mean?
with them, read them or see them, it is the same thing as losing them forever, but with our help, you can restore them. Iow did this happen? Specially for you, on our server was generated the secret key pair RSA-2048 - public and private. If your files were encrypted with the public key, which has been transferred to your computer via the Internet. Decrypting of your files is only possible with the help of the private key and decrypt program, which is on our secret server. What do I do? Was, if you do not take the necessary measures for the specified time then the conditions for obtaining the private key will be changed. f you really value your data, then we suggest you do not waste valuable time searching for other solutions because they do not exist. For more specific instructions, please visit your personal home page, there are a few different addresses pointing to your page below: 1. <u>SSERVICE_WEB_15%</u> 2. <u>SSERVICE_WEB_256</u> 3. <u>SSERVICE_WEB_256</u> 3. <u>SSERVICE_WEB_356</u> 4. <u>SSERVICE_WEB_356</u> 4. <u>SSERVICE_WEB_356</u> 4. <u>SSERVICE_WEB_256</u> 5. 1. <u>Download and install tor-browser: http://www.torproject.org/projects/horbrowser.html.en</u> 2. <u>After a successful installation, nun the browser and wait for initialization. 3. <u>Sype in the addresses are not available, follow these steps:</u> 4. Follow the instructions on the site. MPORTANT INFORMATION: Your Personal PAGE: <u>%SERVICE_WEB_156</u> Your Personal PAGE: <u>%SERVICE_WEB_156</u> Your Personal PAGE: <u>%SERVICE_WEB_156</u> Your Personal PAGE: <u>%SERVICE_TOR56</u></u>	This means t	hat the structure and data within your files have been irrevocably changed, you will not be able to work
two did this happen? Expecially for you, on our server was generated the secret key pair RSA-2048 - public and private. All your files were encrypted with the public key, which has been transferred to your computer via the internet. Decrypting of your files is only possible with the help of the private key and decrypt program, which is on our secret server. What do i do? Was, if you do not take the necessary measures for the specified time then the conditions for obtaining the private key will be changed. If you really value your data, then we suggest you do not waste valuable time searching for other solutions because they do not exist. For more specific instructions, please visit your personal home page, there are a few different addresses pointing to your page below: 1.%SERVICE_WEB_1% 2.%SERVICE_WEB_2% 3.%SERVICE_WEB_4% If for some reasons the addresses are not available, follow these steps: 1. Download and install tor-browser: http://www.insproject.org/projects/horbrowser.html.en 2. After a successful installation, nun the browser and wait for initialization. 3. Type in the address bar: %SERVICE_TOR% 4. Follow the instructions on the site. MPORTANT INFORMATION: Your Personal PAGE: %SERVICE_TOR%	vith them, re	ad them or see them, it is the same thing as losing them forever, but with our help, you can restore them.
Specially for you, on our server was generated the secret key pair RSA-2048 - public and private. We your files were encrypted with the public key, which has been transferred to your computer via the internet. Decrypting of your files is only possible with the help of the private key and decrypt program, which is on our secret server. What do I do? Nas, if you do not take the necessary measures for the specified time then the conditions for obtaining the private key will be changed. If you really value your data, then we suggest you do not waste valuable time searching for other solutions because they do not exist. For more specific instructions, please visit your personal home page, there are a few different addresses pointing to your page below: 1.%SERVICE_WEB_1% 2.%SERVICE_WEB_2% 3.%SERVICE_WEB_2% 3.%SERVICE_WEB_2% 3.%SERVICE_WEB_3% 4.%SERVICE_TOR% 4. Foliow the instructions on the site. MPORTANT INFORMATION: Your Personal PAGE: %SERVICE_WEB_1% Your Personal PAGE: %SERVICE_WEB_1% Your Personal PAGE: %SERVICE_TOR%	low did thir	s happen?
All your files were encrypted with the public key, which has been transferred to your computer via the Internet. Decrypting of your files is only possible with the help of the private key and decrypt program, which is on our secret server. Nhat do I do? Vas, if you do not take the necessary measures for the specified time then the conditions for obtaining the private key will be changed. f you really value your data, then we suggest you do not waste valuable time searching for other solutions because they do not exist. For more specific instructions, please visit your personal home page, there are a few different addresses pointing to your page below: 1.%SERVICE_WEB_1% 2.%SERVICE_WEB_2% 3.%SERVICE_WEB_2% 3.%SERVICE_WEB_3% 4.%SERVICE_WEB_4% If for some reasons the addresses are not available, follow these steps: 1. Download and install tor-browser: http://www.iorproject.org/projects/lorbrowser.html.en 2. After a successful installation, run the browser and wait for initialization. 3. Type in the address bar: %SERVICE_TOR% 4. Follow the instructions on the site. MPORTANT INFORMATION: Your Personal PAGE: %SERVICE_TOR%	Especially fo	r you, on our server was generated the secret key pair RSA-2048 - public and private.
Decrypting of your files is only possible with the help of the private key and decrypt program, which is on our secret server. What do I do? Was, if you do not take the necessary measures for the specified time then the conditions for obtaining the private key will be changed. I you really value your data, then we suggest you do not waste valuable time searching for other solutions because they do not exist. For more specific instructions, please visit your personal home page, there are a few different addresses pointing to your page below: 1.%SERVICE_WEB_1% 2.%SERVICE_WEB_2% 3.%SERVICE_WEB_3% 4.%SERVICE_WEB_3% 4.%SERVICE_WEB_4% If for some reasons the addresses are not available, follow these steps: 1. Download and install tor-browser. <u>http://www.torproject.org/projects/forbrowser.html.en</u> 2. After a successful installation, run the browser and wait for initialization. 3. Type in the address bar: %SERVICE_TOR% 4. Follow the instructions on the site. MPORTANT INFORMATION: Your Personal PAGE: %SERVICE_TOR%	All your files	were encrypted with the public key, which has been transferred to your computer via the Internet.
What do I do? Vas, if you do not take the necessary measures for the specified time then the conditions for obtaining the private key will be changed. f you really value your data, then we suggest you do not waste valuable time searching for other solutions because they do not exist. For more specific instructions, please visit your personal home page, there are a few different addresses pointing to your page below: 1.% SERVICE_WEB_1% 2.%SERVICE_WEB_2% 3.%SERVICE_WEB_4% If for some reasons the addresses are not available, follow these steps: 1. Download and install tor-browser: http://www.torproject.org/projects/torbrowser.html.en 2. After a successful installation, run the browser and wait for initialization. 3. Type in the address bar: % SERVICE_TOR% 4. Follow the instructions on the site. MPORTANT INFORMATION: Your Personal PAGE: % SERVICE_WEB_1% Your Personal PAGE: % SERVICE_TOR%	Jecrypting o	f your files is only possible with the help of the private key and decrypt program, which is on our secret server.
What do I do? Vas, if you do not take the necessary measures for the specified time then the conditions for obtaining the private key will be changed. f you really value your data, then we suggest you do not waste valuable time searching for other solutions because they do not exist. For more specific instructions, please visit your personal home page, there are a few different addresses pointing to your page below: 1.% SERVICE_WEB_1% 2.% SERVICE_WEB_2% 3.% SERVICE_WEB_4% If for some reasons the addresses are not available, follow these steps: 1. Download and install tor-browser: http://www.torproject.org/projects/torbrowser.html.en 2. After a successful installation, run the browser and walt for initialization. 3. Type in the address bar: % SERVICE_TOR% 4. Follow the instructions on the site. MPORTANT INFORMATION: Your Personal PAGE: % SERVICE_WEB_1% Your Personal PAGE: % SERVICE_TOR%		
Vas, if you do not take the necessary measures for the specified time then the conditions for obtaining the private key will be changed. f you really value your data, then we suggest you do not waste valuable time searching for other solutions because they do not exist. For more specific instructions, please visit your personal home page, there are a few different addresses pointing to your page below: 1.%SERVICE_WEB_1% 2.%SERVICE_WEB_2% 3.%SERVICE_WEB_4% If for some reasons the addresses are not available, follow these steps: 1. Download and install tor-browser: <u>http://www.torproject.org/projects/torbrowser.html.en</u> 2. After a successful installation, nun the browser and wait for initialization. 3. Type in the address bar: %SERVICE_TOR% 4. Follow the instructions on the site. MPORTANT INFORMATION: Your Personal PAGE: %SERVICE_WEB_1% Your Personal PAGE: %SERVICE_TOR%	What do I do	p?
f you really value your data, then we suggest you do not waste valuable time searching for other solutions because they do not exist. For more specific instructions, please visit your personal home page, there are a few different addresses pointing to your page below: 1.%SERVICE_WEB_1% 2.%SERVICE_WEB_2% 3.%SERVICE_WEB_2% 4.%SSERVICE_WEB_4% If for some reasons the addresses are not available, follow these steps: 1. Download and install tor-browser: http://www.torproject.org/projects/torbrowser.html.en 2. After a successful installation, nun the browser and wait for initialization. 3. Type in the address bar: %SERVICE_TOR% 4. Follow the instructions on the site. MPORTANT INFORMATION: Your Personal PAGE: %SERVICE_WEB_1% Your Personal PAGE(using TOR): %SERVICE_TOR%	Mas, if you d	o not take the necessary measures for the specified time then the conditions for obtaining the private key will be changed.
For more specific instructions, please visit your personal home page, there are a few different addresses pointing to your page below: 1.%SERVICE_WEB_1% 2.%SERVICE_WEB_2% 3.%SERVICE_WEB_3% 4.%SERVICE_WEB_3% 4.%SERVICE_WEB_4% If for some reasons the addresses are not available, follow these steps: 1. Download and install tor-browser: http://www.iorproject.org/projects/lorbrowser.html.en 2. After a successful installation, run the browser and wait for initialization. 3. Type in the address bar: %SERVICE_TOR% 4. Follow the instructions on the site. MPORTANT INFORMATION: Your Personal PAGE: %SERVICE_WEB_1% Your Personal PAGE(using TOR): %SERVICE_TOR%	f you really v	alue your data, then we suggest you do not waste valuable time searching for other solutions because they do not exist.
For more specific instructions, please visit your personal home page, there are a few different addresses pointing to your page below: 1.%SERVICE_WEB_2% 3.%SERVICE_WEB_2% 4.%SERVICE_WEB_4% If for some reasons the addresses are not available, follow these steps: 1. Download and install tor-browser: http://www.torproject.org/projects/torbrowser.html.en 2. After a successful installation, run the browser and walt for initialization. 3. Type in the address bar: %SERVICE_TOR% 4. Follow the instructions on the site. MPORTANT INFORMATION: Your Personal PAGE: %SERVICE_TOR%		
	r or more sp	vectic instructions, please visit your personal nome page, there are a tew different addresses pointing to your page below:
2.%SERVICE_WEB_2% 3.%SERVICE_WEB_3% 4.%SERVICE_WEB_4% If for some reasons the addresses are not available, follow these steps: 1. Download and install tor-browser: <u>http://www.torproject.org/projects/torbrowser.html.en</u> 2. After a successful installation, run the browser and walt for initialization. 3. Type in the address bar: %SERVICE_TOR% 4. Follow the instructions on the site. MPORTANT INFORMATION: Your Personal PAGE: <u>%SERVICE_WEB_1%</u> Your Personal PAGE(using TOR): <u>%SERVICE_TOR%</u>	1.%SERVIC	E_WEB_1%
3.% SERVICE_WEB_4% If for some reasons the addresses are not available, follow these steps: 1. Download and install tor-browser: <u>http://www.torproject.org/projects/torbrowser.html.en</u> 2. After a successful installation, run the browser and wait for initialization. 3. Type in the address bar: % SERVICE_TOR% 4. Follow the instructions on the site. MPORTANT INFORMATION: Your Personal PAGE: % SERVICE_TOR% Your Personal PAGE(using TOR): % SERVICE_TOR%	2.% SERVIC	:E_WEB_2%
4.%SERVICE_WEB_4% If for some reasons the addresses are not available, follow these steps: Download and install for-browser. <u>http://www.torproject.org/projects/forbrowser.html.en</u> After a successful installation, run the browser and wait for initialization. After a successful installation, run the browser and wait for initialization. Type in the address bar: %SERVICE_TOR% Follow the instructions on the site. MPORTANT INFORMATION: Your Personal PAGE: <u>%SERVICE_WEB_1%</u> Your Personal PAGE(using TOR): <u>%SERVICE_TOR%</u>	3.%SERVIC	E_WEB_3%
If for some reasons the addresses are not available, follow these steps: 1. Download and install tor-browser. <u>http://www.torproject.org/projects/forbrowser.html.en</u> 2. After a successful installation, run the browser and wait for initialization. 3. Type in the address bar: %SERVICE_TOR% 4. Follow the instructions on the site. MPORTANT INFORMATION: Your Personal PAGE: <u>%SERVICE_WEB_1%</u> Your Personal PAGE(using TOR): <u>%SERVICE_TOR%</u>	4.%SERVIC	:E_WEB_4%
It for some reasons the addresses are not available, follow these steps: Download and install tor-browser. <u>http://www.torproject.org/projects/forbrowser.html.en</u> After a successful installation, run the browser and walt for initialization. Type in the address bar: %SERVICE_TOR% 4. Follow the instructions on the site. MPORTANT INFORMATION: Your Personal PAGE: <u>%SERVICE_WEB_1%</u> Your Personal PAGE(using TOR): <u>%SERVICE_TOR%</u>		the address of a state that the state state
1. Download and install tor-browser: <u>http://www.torproject.org/projects/torbrowser.html.en</u> 2. After a successful installation, run the browser and walt for initialization. 3. Type in the address bar: % SERVICE_TOR% 4. Follow the instructions on the site. MPORTANT INFORMATION: Your Personal PAGE: <u>% SERVICE_WEB_1%</u> Your Personal PAGE(using TOR): % SERVICE_TOR%	If for some r	easons the addresses are not available, follow these steps.
2. After a successful installation, run the browser and wait for initialization. 3. Type in the address bar: % SERVICE_TOR% 4. Follow the instructions on the site. MPORTANT INFORMATION: Your Personal PAGE: % SERVICE_WEB_1% Your Personal PAGE(using TOR): % SERVICE_TOR%	1. Download	and install tor-browser: http://www.torproject.org/projects/torbrowser.html.en
3. Type in the address bar: % SERVICE_TOR% 4. Follow the instructions on the site. MPORTANT INFORMATION: Your Personal PAGE: % SERVICE_WEB_1% Your Personal PAGE(using TOR): % SERVICE_TOR%	After a su	ccessful installation, run the browser and wait for initialization.
4. Follow the instructions on the site. MPORTANT INFORMATION: Your Personal PAGE: <u>%SERVICE_WEB_1%</u> Your Personal PAGE(using TOR): <u>%SERVICE_TOR%</u>	3. Type in th	ve address bar: % SERVICE_TOR%
MPORTANT INFORMATION: Your Personal PAGE: <u>%SERVICE_WEB_1%</u> Your Personal PAGE(using TOR): %SERVICE_TOR%	Follow the	e instructions on the site.
MPORTANT INFORMATION: Your Personal PAGE: <u>% SERVICE_WEB_1%</u> Your Personal PAGE(using TOR): % SERVICE_TOR%		
MPORTANT INFORMATION: Your Personal PAGE: <u>%SERVICE_WEB_1%</u> Your Personal PAGE(using TOR): %SERVICE_TOR%		
Your Personal PAGE: <u>% SERVICE_WEB_1%</u> Your Personal PAGE(using TOR): % SERVICE_TOR%		(INFORMATION:
Your Personal PAGE(using TOR): %SERVICE_TOR%	MPORTAN	
	MPORTANT Your Person	al PAGE: <u>% SERVICE_WEB_1%</u>

Once the note is decompressed, the HTML fields will be populated with multiple TOR addresses at subroutine sub_00414160()). The note is stored in memory then follows a few more checks before the malware sends its first C2 *POST* request. Stepping into SendRequestToC2 which is located at 0x00416A50, the first thing we notice is a buffer being allocated 60 bytes of memory.

.text:00416A77 018 push 3Ch.text:00416A79 01C call AllocateSetMemory.text:00416A7E 01C add esp, 4.text:00416A81 018 mov [ebp+campaign_str], eax

All this information will eventually help us write a proper <u>fake C2 server</u> that will allow us to communicate with the ransomware since CryptoWall's I2P servers are no longer active. Around address 0x004052E0, which we labeled <u>EncryptData_SendToC2</u> will be responsible for taking our generated campaign string and sending it as an initial ping.

c4_encrypted_campaign_str = AllocateSetMemory(60); if (rc4_encrypted_campaign_str) ł if (GenerateCampaignIDAsString(7, rc4_encrypted_campaign_str + 9, rc4_encrypted_campaign_str + 10)) ConcatString(rc4_encrypted_campaign_str + 9, rc4_encrypted_campaign_str + 10, 1); if (EncryptData_SendToC2(rc4_encrypted_campaign_str))

If you set a breakpoint at this function, you can see what the parameter contains: {1|crypt1|C6B359277232C8E248AFD89C98E96D65} . Once inside this module, you'll notice three key functions; one responsible for byte swapping, a key scheduling algorithm, and the other doing the actual encryption. The generated RC4 encryption will end up as a hash string:

85b088216433863bdb490295d5bd997b35998c027ed600c24d05a55cea4cb3deafdf4161e6781d2cd9aa24

Command & Control Communication

The malware sets itself up for a POST request to its I2P addresses that cycle between proxy1-1-1.i2p & proxy2-2-2.i2p. The way this is done is by using the function at 0x0040B880 to generate a random seed based on epoch time, and use that to create a string that ranges from 11 to 16 bytes. This PRNG (Pseudo-Random Number Generator) string will be used as the <u>POST request's URI</u> and as the key used in the byte swapping function before the RC4 encryption.



To give us an example, if our generated string results in tfuzxqh6wf7mng, then after the function call, that string will turn into 67ffghmnqtuwxz. That string gets used for a 256-generated key scheduling algorithm, and the POST request (I.E., http://proxy1-1-1.i2p/67ffghmnqtuwxz). You can find the reverse engineered algorithm <u>here</u>.

030 mov	[ebp+data_out], 0
030 mov	[ebp+size_of_hash], 0
030 lea	edx, [ebp+size of hash]
030 push	edx
034 lea	eax, [ebp+data_out]
034 push	eax
038 mov	ecx, [ebp+some_key_generated]
038 push	ecx
03C mov	edx, [ebp+ <mark>rc4 encrypted campaign str</mark>]
03C mov	eax, [edx+28h]
03C push	eax
040 mov	ecx, [ebp+ <mark>rc4 encrypted campaign str</mark>]
040 mov	edx, [ecx+24h]
040 push	edx
044 call	RC4_EncryptCampaignID
044 add	esp, 14h
030 test	eax, eax
030 jz	loc_405530

The next part will take this byte swapped key, then RC4 encrypt some campaign information that the malware has gathered, which unencrypted, will look like this:

```
{1|crypt1|C6B359277232C8E248AFD89C98E96D65|0|2|1||55.59.84.254}
```

This blob consists of the campaign ID, an MD5 hashed unique computer identifier, a CUUID, and the victims public IP address. After preparation of this campaign string, the ransomware will begin to resolve the two I2P addresses. Once CryptoWall sends its first ping to the C2 server, the malware expects back an RC4 encrypted string, which will contain a public key used to encrypt all the files on disk. The malware has the ability to decrypt this string using the same RC4 algorithm from earlier, and will parse the info from this block:

{216|1pai7ycr7jxqkilp.onion|[pub_key]|US|[unique_id]}. The onion route is for the ransom note, and is a personalized route that the victim can enter using a TOR browser. The site most likely contains further instructions on how to pay the ransom.

Since the C2 servers are no longer active; in order to actually know what our fake C2 server **should** send back to the malware; the parser logic had to be carefully dissected which is located at 0x00405203.



In this block, the malware decrypts the data it received from the C2 server. Once decrypted, it stores the first byte in ECX and compares hex value to 0x7B (char: '{'). Tracing this function call to the return value, the string returned back will remove brackets from start to end. At memory address 0x00404E69, a DWORD pointer at eax+2ch holds our newly decrypted and somewhat parsed string, that will be checked for a length greater than 0. If the buffer holds weight, we move on over to the final processing of this string routine at 0x00404B00, that I dubbed ParseC2Data(). This function takes four parameters, char* datain, int datain_size, char *dataout, int dataout_size. The first blob on datain data gets parsed from the first 0x7C (char: '|') and extracts the victim id.

victim_id = GetXBytesFromC2Data(decrypted_block_data_from_c2, &hex_7c, &ptr_to_data_out);

ptr_to_data_out and EAX will now hold an ID number of 216 (we got that number since we placed it there in our fake C2). The next block of code will finish the rest of the data:

```
while ( victim_id ){
                       if ( CopyMemoryToAnotherLocation(&some_buffer_to_copy_too,
                          CopyBlocksofMemory(victim_id,
8 * idx + 8) )
                 {
&some_buffer_to_copy_too[2 * idx + 1],
&some_buffer_to_copy_too[2 * idx]);
                                            ++idx;
                                                          if ( ptr_to_data_out )
             for ( i = 0; *(i + ptr_to_data_out) == 0x7C; ++i )
{
                                                                             {
if (
                   CopyMemoryToAnotherLocation(&some_buffer_to_copy_too,
8 * idx + 8) )
                                                   ++v9;
                                                                            ++idx;
                               {
                                   victim_id = GetXBytesFromC2Data(0, &hex_7c_0,
}
              }
                        }
                              }
&ptr_to_data_out);
                      ++v5;
                                ++v9;}
```

What's happening here is that by every iteration of the character (')' we grab the next chunk of data and place it in memory into some type structure. The data jumps X amount of times per loop until it reaches the last $0 \times 7C$ byte. It will loop a total of four times. After this function returns, dataout will contain a pointer in memory to this local type, which we reversed to look like this:

```
struct _C2ResponseData{ int victim_id; char *onion_route; const char*
szPemPubKey; char country_code[2]; char unique_id[4];};
```

Shortly after, there is a check to make sure the victim id generated is no greater than 0x3E8 or that it is not an unsigned value.

```
value_of_index = CheckID(*(*parsed_data_out->victim_id));if ( value_of_index > 0x3E8
|| value_of_index == 0xFFFFFFFF ) value_of_index = 0x78;
```

I believe certain malware will often perform these checks throughout the parsing of the C2 response server to make sure the data being fed back is authentic. Over at 0x00404F35, there is another check to see how many times it tried to reach the command server. If the check reaches exactly 3 times then it will move to check if the onion route is valid; all CryptoWall variants hardcode the first string index with ascii '1'. If it does not start with this number, then it will try to reach back again for a different payload. The other anti-tamper check it makes for the onion route is a CRC32 hash against the payload, if the compressed route does not equal 0x63680E35, the malware will try one last time to compare against the DWORD value of 0x30BBB749. The variant has two hardcoded 256 byte arrays to which it compares the encrypted values against. Brute-forcing can take a long time but is possible with a python script that I made here. The checksum is quite simple, it will take each letter of the site string and logical-XOR against an unsigned value:

tmp = ord(site[i])) ^ (ret_value & 0xfffff)

It will take the tmp value and use it as an index in the hardcoded byte array to perform another logical-XOR against :

```
ret_value = bytes_array[tmp*4:(tmp*4)+4] ^ (0xFFFFFFF >> 8)
```

The return value then gets <u>inverted</u> giving us a 4 byte hash to verify against. Now the malware moves on over to the main thread responsible for encrypting the victims files at 0x00412988. The first function call in this thread is from CryptAcquireContextW, and

that will acquire a handle to a particular key container within a CSP. **16** bytes will then be allocated to the stack using VirtualAlloc; which will be the buffer to the original key.

```
isDecompressed = CreateTextForRansomwareNote(0, 0, 0);if ( !isRequestSuccess ||
!isDecompressed ){ remaining_c2_data = 0; while ( 1 ) { isRequestSuccess =
SecondRequestToC2(&rsa_key, &rsa_key_size,
&remaining_c2_data); if ( isRequestSuccess ) break;
sleep(0x1388u);}
```

Once the text for the ransom note is decompressed, CryptoWall will place this note as an HTML, PNG, and TXT file inside of every directory the virus went through to encrypt documents. After this point, it will go through another round of requests to the I2P C2 servers to request another RSA 2048-bit public key. This key will be the one used for encryption. This strain will do a number of particular hardcoded hash checks on the data it gets back from the C2.

Decoding the Key

CryptoWall will use basic Win32 Crypto functions like CryptStringToBinaryA, CryptDecodeObjectEx, & CryptImportPublicKeyInfo to decode the RSA key returned. Then it will import the public key information into the provider which then returns a handle of the public key. After importing is finished, all stored data will go into a local type structure like this:

struct _KeyData{ char *key; int key_size; BYTE *hash_data_1; BYTE *hash_data_2;};// Gets used here at 0x00412B8Cif (ImportKey_And_EncryptKey(cryptContext, rsa_key, rsa_key_size, OriginalKey->key, &OriginalKey->key_size, &OriginalKey->hash_data_1, &OriginalKey->hash_data_2)){

The next actions the malware takes is pretty basic for ransomware.. it will loop through every available drive, and use **GetDriveTypeW** to determine whether a disk drive is a removable, fixed, CD-ROM, RAM disk, or network drive. In our case, the C drive is the only open drive which falls under the category of **DRIVE_FIXED**. CryptoWall will only check if the drive is CD-ROM because it will not try to spread in that case.

.text:00412C1B mov ecx, [ebp+driver_letter].text:00412C1E push ecx.text:00412C1F call GetDriveTypeW.text:00412C2C cmp eax, 5.text:00412C2F jz skip_drive

EAX holds the integer value returned from the function call which represents the type of drive associated with that number (5 == **DRIVE_CDROM**). You can find the documentation <u>here</u>.

The exciting part is near as we are about to head over to where the malware duplicates the key it retrieved from our <u>fake C2 server</u> at address $0\times00412C7A$. What is happening here is pretty straight forward, and we can show in pseudo-code:

if (OriginalKey) DuplicatedKey = HeapAlloc(16) if (DuplicatedKey) CryptDuplicateKey(OriginalKey, 0, 0, DuplicatedKey) memcpy(DuplicatedKey, OriginalKey, OrignalKey_size) CryptDestroyKey(OriginalKey)

Essentially CryptDuplicateKey is making an exact copy of a key and the state of the key. The DuplicatedKey variable ends up becoming a struct as we can see after the function call at 0x00412C7A , it gets used to store volume information about the drive its currently infecting.

```
GetVolumeInformation(driver_letter, DuplicatedKey + 20);if (
MoveDriverLetterToDupKeyStruct(driver_letter, (DuplicatedKey + 16), 0) { ...
```

That is why 24 bytes was used to allocate to the heap when creating this variable instead of 16. Now we can define our struct from what we know so far:

```
struct _DupKey{ const char *key; int key_size; DWORD unknown1; DWORD
unknown2; char *drive_letter; LPDWORD lpVolumeSerialNumber; DWORD unknown3;};//
Now our code looks cleaner from aboveGetVolumeInformation(driver_letter,
&DuplicatedKey->lpVolumeSerialNumber);if (
MoveDriverLetterToDupKeyStruct(driver_letter, &DuplicatedKey->drive_letter, 0) {
...
```

Encrypting of Files

After the malware is finished storing all pertinent information regarding how and where it will do its encryption, CryptoWall moves forward to the main encryption loop at 0x00416780.



Encryption Loop Control Flow Graph

As we can see, the control flow graph is fairly long in this subroutine, but nothing out of the ordinary when it comes to ransomware. A lot has to be done before encrypting files. At the start of this function, we see an immediate call to HeapAlloc to allocate 260 bytes of memory. We can automatically assume this will be used to store the file's absolute path, as Windows OS only allows a max of 260 bytes. Upon success, there is also an allocation of virtual memory with a size of 592 bytes that will later be used as the file buffer contents. Then the API call FindFirstFilew uses this newly allocated buffer to store the first filename found on system. The pseudo-code below will explain the flow:

```
lpFileName = Allocate260BlockOfMemory(); // HeapAllocif ( lpFileName ){ (*(wcscpy +
292))(lpFileName, driver_letter); ... lpFindFileData = AllocateSetMemory(592); //
VirtualAlloc if ( lpFindFileData ) { hFile = (*(FindFirstFileW + 504))
(lpFileName, lpFindFileData); if ( hFile != 0xFFFFFFF ) { v29 = 0;
do { // Continue down to further file actions
```

Before the malware opens up the first victim file, it needs to make sure the file and file extension themselves are not part of their hardcoded blacklist of bytes. It does this check using a simple CRC-32 hash check. It will take the filename, and extension; compress it down to a DWORD, then compare that DWORD to a list of bytes that live in the .data section.

.data:00420F90			; int comp	ressed_e	ktensi	ons[313]
.data:00420F90	89 4E 33	EA	compressed	_extensio	ons dd	OEA	334E89h
.data:00420F90				_			
.data:00420F94	2A			db	2Ah		
.data:00420F95	DB			db	ODBh	; Û	
.data:00420F96	57			db	57h		
.data:00420F97				db	74h	; t	
.data:00420F98	90			db	90h		
.data:00420F99	8A			db	8Ah		
.data:00420F9A	5E			db	5Eh		
.data:00420F9B	ED			db	OEDh	; í	
.data:00420F9C	49			db	49h	; I	
.data:00420F9D	E6			db	0E6h		
.data:00420F9E	58			db	58h	; X	
.data:00420F9F	8E			db	8Eh		
.data:00420FA0	4E			db	4Eh		
.data:00420FA1	СВ			db	OCBh		
.data:00420FA2	EO			db	OEOh		
.data:00420FA3	69			db	69h		
.data:00420FA4	D8			db	0D8h		
.data:00420FA5	FB			db	OFBh	; û	
.data:00420FA6	E7			db	OE7h	;ç	

To see how the algorithm works, I reversed it to python code, and wrote my own file checker.

→ python tor_site_checksum_finder.py --check-file-ext "dll"

[!] Searching PE sections for compressed .data

[!] Searching PE sections for compressed extension .data

[-] '.dll' is not a valid file extension for Cryptowall

→ python tor_site_checksum_finder.py --check-file-ext "py"

- [!] Searching PE sections for compressed .data
- [!] Searching PE sections for compressed extension .data

[+] '.py' is a valid file extension for Cryptowall

Now we can easily tell what type of files CryptoWall will attack. Obvious extensions like .dll, .exe, and .sys is a very common file type for ransomware to avoid.



If the file passes these two checks, then it moves on over to the last part of the equation; the actual encryption located at 0x00412260. We can skip the first few function calls as they are not pertinent to what is about to happen. If you take a look at address 0x00412358, there is a subroutine that takes in three parameters; a file handle, our DuplicateKeyStruct, and a file size. Stepping into the function, we can immediately tell what is happening:

```
if(ReadFileA(hFile, lpBuffer, DuplicateKeyStruct->file_hash_size,
&lpNumberOfBytesRead, 0) && lpNumberOfBytesRead) == DuplicateKeyStruct-
>file_hash_size{ if(memcmp(lpBuffer, DuplicateKeyStruct->file_hash,
DuplicateKeyStruct->file_hash_size)) { isCompare = 1; }}
```

The pseudo-code is telling us that if an MD5 hash of the file is present in the header, then its already been encrypted. If this function returns **isCompared** to be true, then CryptoWall moves on to another file and will leave this one alone. If it returns false from the **Compare16ByteHeader()** function call, the malware will append to the file's extension by using a simple algorithm to generate a three lettered string to place at the end. The generation takes a timestamp, uses it as a seed, and takes that seed to then mod the first three bytes by 26 then added to 97.

*(v8 + 2 * i) = DataSizeBasedOnSeed(0, 0x3E8u) % 26 + 97;

This is essentially a rotation cipher, where you have a numerical variable checked by a modulate to ensure it doesn't go past alphanumeric values, then the addition to 97 rotates the ordinal 45 times. As an example, if we have the letter 'A', then after this cipher, it ends up becoming an 'n'. In conclusion, if the victim file is named hello.py, this subroutine will rename it to hello.py.3xy.

Next, around address 0x004123F0, the generation of an AES-256 key begins with another call to Win32's CryptAcquireContextW. The phProv handler gets passed over to be used in CryptGenKey and CryptGetKeyParam.

if (CryptGenKey(hProv, 0x6610, 1, &hKey)): pbData_1 = 0; pdwDataLen_1 = 4; if (CryptGetKeyParam(hKey, 8, &pbData_1, &pdwDataLen_1, 0, 4) The hexadecimal value of 0x6610 shown above tells us that the generated key is going to be AES-256 as seen in <u>MS-DOCS</u>. Once the hKey address to which the function copies the handle of the newly generated key is populated, CryptGetKeyParam will be used to make the key and transfer it into pbData ; a pointer to a buffer that receives the data. One last call in this function we labeled as GenerateAESKey() gets called which is CryptExportKey. This will take the handle to the key to be exported and pass it the function, and the function returns a key BLOB. The second parameter of the GenerateAESKey() will hold the aes_key.



The next call is one of the most important ones to understand how eventually we can decrypt the files that CryptoWall infected. EncryptAESKey() uses the pointer to DuplicateKeyStruct->rsa_key to encrypt our AES key into a 256 byte blob. Exploring inside this function call is fairly simple; it uses CryptDuplicateKey and CryptEncrypt to take our public RSA 2048-bit key from earlier, our newly generated AES key to duplicate both keys to save for later, and encrypt the buffer. The fifth parameter is our data out in this case and once the function returns, what we labeled as encrypted_AESkey_buffer will hold our RSA encrypted key.

At around address <u>004124A5</u>, you will see two calls to <u>WriteFileA</u>. The first call will move the <u>16</u> byte MD5 hash at the top of the victim file, and the second call will write out the <u>256</u> bytes of encrypted key buffer right below the hash.

	•								victim	_file.txt.3	Sxy		
000	91323586	F370D899	CA31FCF3	DØA2AF22	9F0F6F9E	B87E6438	Ø3FBB848	C8EE612D	DF43ABFC	93D64B83	8F00A389	85275F2A	[ë25ÜÜpÿô 1,Ū-¢Ø"¦ù oû∏~d8 '∏H»Óa-flC´,ì÷KÉè £âÖ'_*
030	7A7B649E	EEE5D3FF	ØCC867E7	F71548B9	286133FD	Ø63172AD	FA581C5C	Ø18FBADØ	2EAA5AØ3	CEØA666E	8A6F6561	9D24BFF8	z{dûOA" »gA" Hπ(a3″ 1r≠ X ∖ è∫™Z Œ fnäoeaù\$ø⁻
060	9D9247DA	BC7CD221	56171E70	D50D4C66	D6275A1F	835E7056	Ø618631A	FC979C28	EC87BA62	C7787667	9E68D1EE	66B02593	ùiG/° "!V p'Lf÷'Z É^pV c _óú(Ïá∫b«xvgûh-Óf∞%ì
090	42534420	322D436C	61757365	204C6963	656E7365	ØAØA436F	70797269	67687420	28632920	32303138	2C204A6F	686E204D	BSD 2-Clause License Copyright (c) 2018, John M
000	634D6173	7465720A	416C6C20	72696768	74732072	65736572	7665642E	ØAØA5265	64697374	72696275	74696F6E	20616E64	cMaster All rights reserved. Redistribution and
0F0	20757365	20696E20	736F7572	63652061	6E642062	696E6172	7920666F	726D732C	20776974	68206F72	20776974	686F7574	use in source and binary forms, with or without
120	ØA6D6F64	69666963	6174696F	6E2C2061	72652070	65726D69	74746564	2070726F	76696465	64207468	61742074	68652066	modification, are permitted provided that the f
150	6F6C6C6F	77696E67	20636F6E	64697469	6F6E7320	61726520	6D65743A	0A0A2A20	52656469	73747269	62757469	6F6E7320	ollowing conditions are met: * Redistributions
180	6F662073	6F757263	6520636F	6465206D	75737420	72657461	696E2074	68652061	626F7665	20636F70	79726967	6874206E	of source code must retain the above copyright n
180	6F746963	652C2074	6869730A	20206069	7374206F	6620636F	6E646974	696F6E73	20616E64	20746865	20666F6C	6C6F7769	otice, this list of conditions and the followi
1E0	6E672064	69736360	61696D65	722E0A0A	ZA205265	64697374	72696275	74696F6E	7320696E	2062696E	61727920	666F726D	ng disclaimer. * Redistributions in binary form
210	20607573	74207265	70726F64	75636520	74686520	61626F76	6520636F	70797269	67687420	6E6F7469	63652CØA	20207468	must reproduce the above copyright notice, th
240	69732060	69/3/420	6F662063	6F6E6469	74696F6E	7320616E	64207468	6520666F	6C6C6F77	696E6720	64697363	6C61696D	is list of conditions and the following disclaim
2/0	65722069	6E207468	6520646F	63756065	6E/461/4	696F6E0A	2020616E	64216172	20617468	6572206D	61/465/2	69616073	er in the documentation and/or other materials
2A0	20/0/261	76696465	64207769	/46820/4	68652064	69/3/4/2	69627574	090F6E2E	0A0A5448	49532053	41465457	41524520	provided with the distribution. THIS SUFTWARE
200	49532050	52415649	44454420	42592054	48452043	41505952	494/4854	20484F4C	44455253	20414E44	20434142	54524942	IS PROVIDED BY THE COPTRIGHT HOLDERS AND CONTRIB
300	33344132	33202241	53204953	220A414E	4420414E	59204558	50524553	53204152	20494050	40494544	20574152	52414E54	UTURS "AS IS" AND ANT EXPRESS OR IMPLIED WARRANT
350	49400020	20494643	40554449	46304045	42000420	46465420	40404049	59454420	544F2C20	5448450A	4940504C	49454420	MADDANTIES OF MEDGUANTADILITY AND ETTNESS FOR A
200	57415252	41455449	40002045	40204045	52454641	45344142	49404954	3920414E	44204049	344E4333	3320404F	52204120	MARKANITES OF MERCHANIABILITT AND FITNESS FOR A
300	30413234	4943334C	41522050	50523045	33432041	32430A44	49004052	41494045	442E2049	46204646	20400040	46542055	HALL THE CODVIDENT HOLDER OF CONTRIBUTORS RELITA
360	40414040	46455220	20454F50 414E5020	39324947	40342040	46464445	52454354	20434646	34324942	33344F32	20205350	45434941	RIE EOR ANY DIRECT INDIRECT INCIDENTAL SPECIA
420	40202045	58454050	41423320	20204552	20434545	53455155	45455449	41400444	41404147	45532028	A04E434C	5544941	L EXEMPLARY OF CONSEQUENTIAL DAMAGES (INCLUDIN
450	472(2042	5554204F	4E542040	49404954	45442054	4F2C2050	524E4355	52454D45	4E54204E	46205355	42535449	54555445	C BUT NOT I INITED TO PROCUREMENT OF SUBSTITUTE
480	20474545	4453204E	52045345	52564943	45533R20	4C4F5353	204F4620	55534520	20444154	412C204F	52205052	4F464954	GOODS OR SERVICES: LOSS OF USE DATA OR PROFIT
4R0	533R204F	52204255	53494F45	53532049	455335620	52555054	494F4F29	20484F57	45564552	04434155	53454420	41464334	S: OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
4F0	4F4F2041	4F592054	48454F52	59204F46	204(4941	42494(49	54592(20	57484554	48455220	494F2043	4F4F5452	41435420	ON ANY THEORY OF LIABLITY, WHETHER IN CONTRACT.
510	20535452	49435420	40494142	494(4954	592C044F	5220544F	52542028	494F434C	5544494F	47204F45	474(4947	454F4345	STRICT LIABLITY, OR TORT (INCLUDING NEGLIGENCE
540	204F5220	4F544845	52574953	45292041	52495349	4F472049	4F20414F	59205741	59204E55	54204F46	20544845	20555345	OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
570	ØA4F4620	54484953	20534F46	54574152	452C2045	56454E20	49462041	44564953	4544204F	46205448	4520504F	53534942	OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIB
5A0	494C4954	59204F46	20535543	48204441	4D414745	2E							ILITY OF SUCH DAMAGE.
												I	
												I	
Sig	ned Int) le, dec) (selec	t some dat	a)								- (+

Screenshot shows 128 byte encrypted key buffer, but it was a copy mistake; Supposed to be 256 bytes of encrypted key text.

The picture above shows what an example file will look like up until this stage of the infection. The plaintext is still intact, but the headers now hold the hash of the file and the encrypted AES key used to encrypt the plaintext in the next phase. **ReadFileA** will shortly get called at 0x0041261B, which will read out everything after the header of the file to start the encryption process.

```
if ( (*(ReadFile + 440))(v50, v51, v52, v53, v54) )
  if (lpNumberOfBytesRead == nNumberOfBytesToRead)
  {
    encrypted_buffer_out = 0;
    encrypted_key_buffer_size_1 = 0;
    if ( DuplicateAESKey_And_Encrypt(
           hKey
           isFinalBlob,
           ppKeyBlob,
           actual_file_contents,
           lpNumberOfBytesRead,
           &encrypted_buffer_out,
           &encrypted_key_buffer_size_1) )
      WriteFile = ReturnFunctionName(
                    hHandle.
                    encrypted_buffer_out,
                    encrypted_key_buffer_size_1,
                    &lpOverlapped,
                    0);
      if ( (*(WriteFile + 444))(hFile, lpBuffer, v58, v59, v60)
        && lpOverlapped == encrypted_key_buffer_size_1 )
      Ł
        counter += lpNumberOfBytesRead;
        isSuccess = 1;
        v107 += encrypted_key_buffer_size_1;
        FlushFileBuffer = ReturnFunctionName(hHandle, v95);
        (*(FlushFileBuffer + 448))(v62);
      FreeVirtualMemory(encrypted_buffer_out);
```

Now that 272 bytes belong to the header, anything after that we can assume is free range for the next function to deal with. We don't really need to deep dive too much into what DuplicateAESKey_And_Encrypt() does as it is pretty self explanatory. The file contents are encrypted using the already generated AES key from above that was passed into the HCRYPTKEY *hKey variable. The sixth parameter of this function is the pointer which will contain the encrypted buffer. At this point the ransomware will replace the plaintext with an encrypted blob, and the AES key is free'd from memory.



Example of a fully encrypted file

After the file is finished being processed, the loop will continue until every allow listed file type on disk is encrypted.

Decrypting Victim Files

Unfortunately in this case, it is only possible to write a decryption algorithm if you know the private key used which is generated on the C2 side. This is going to be a two step process as in order to decrypt the file contents, we need to decrypt the AES key that has been RSA

encrypted.

The <u>fake C2 server</u> I wrote also includes an area where a private key is generated at the same time that the public key is generated. So in my case, all encrypted files on my VM are able to be decrypted.

Side Note: In order to run this C2 server, you have to place the malware's hardcoded I2P addresses in <u>/etc/hosts</u> on Windows. Then make sure the server has started before executing the malware as there will be a lot of initial verification going back and forth between the malware and 'C2' to ensure its legitimacy. Your file should look like this:

127.0.0.1 proxy1-1-1.i2p127.0.0.1 proxy2-2-2.i2p

15:11:52] "[37mPOST /93n14chwb3qpm HTTP/1.1[0m" 200

Another reason why we un the fake C2 server before executing the malware is so we don't end up in some dead lock state. The output from our server will look something like this:

Step by step, the first thing we have to do is write a program that imports the private key file. I used C++ for this portion because for the life of me I could not figure out how to mimic the CryptDecodeObjectEx API call that decodes the key in a X509_ASN_ENCODING and PKCS_7_ASN_ENCODING format. Once you have the key blob from this function, we can use this function as the malware does and call CryptImportKey, but this time it is a private key and not a public key ;). Since the first 16 bytes of the victim file contains the MD5 hash of the unencrypted file, we know we can skip that part and focus on the 256 bytes after that part of the header. The block size is going be 256 bytes and AES offset will be 272, since that will be the last byte needed in the cryptographic equation. Once we get the blob, it is now okay to call CryptDecrypt and print out the 32 byte key blob:

```
if (!CryptDecrypt(hKey, NULL, FALSE, 0, keyBuffer, &bytesRead)) { printf("
[-] CryptDecrypt failed with error 0x%.8X\n", GetLastError()); return
FALSE; } printf("[+] Decrypted AES Key => "); for(int i = 0; i < bytesRead; i++)
{ printf("%02x", keyBuffer[i]); }</pre>
```

You can find the whole script <u>here</u>. Now that we are half way there and we have an AES key, the last thing to do is write a simple python script that will take that key / encrypted file and decrypt all remaining contents of it after the 272nd byte.

```
enc_data_remainder = file_data[272:]cipher = AES.new(aes_key, AES.MODE_ECB)
plaintext = cipher.decrypt(enc_data_remainder)
```

The script to perform this action is in the same folder on <u>Github</u>. If you want to see how the whole thing looks from start to finish, it will go like this:

```
    → decrypt_aes_key.exe priv_key_1.pem loveme.txt
    [+] Initialized crypto provider
    [+] Successfully imported private key from PEM file
    [!] Extracted encrypted AES keys from file
    [+] Decrypted AES Key =>
    08020000106600002000000040b4247954af27637ce4f7fabfe1ccfc6cd55fc724caa840f82848ea4800b3
    [+] Successfully decrypted key from file
    → python decrypt_file.py loveme.txt
```

```
40b4247954af27637ce4f7fabfe1ccfc6cd55fc724caa840f82848ea4800b320[+] Decrypting
file[+] Found hash header => e91049c35401f2b4a1a131bd992df7a6[+] Plaintext from file:
b'"hello world" \r\n\'
```

Conclusion

Overall this was one of the biggest leading cyber threats back in 2013, and the threat actors behind this malicious virus have shown their years of experience when it comes to engineering a ransomware such as this.

Although this ransomware is over 6 years old, it still fascinated me so much to reverse engineer this virus that I wanted to share all the tooling I have wrote for it. Every step of the way their was another challenge to overcome, whether it was knowing what the malware expected the encrypted payload to look like coming back from the C2, figuring out how to decrypt their C2 I2P servers using RC4, decompressing the ransomware note using some hard to mimic LZNT1 algorithm, or even understanding their obscure way of generating domain URI paths... it was all around a gigantic puzzle for a completionist engineer like myself.

Here is the repository that contains all the programs I wrote that helped me research <u>CryptoWall</u>.

Thank you for following along! I hope you enjoyed it as much as I did. If you have any questions on this article or where to find the challenge, please DM me at my Instagram: @hackersclub or Twitter: @ringoware

Happy Hunting :)