KeyBoy Network Communication Encoding

This document is a technical supplement to the report [It's Parliamentary: KeyBoy and the targeting of the Tibetan Community]

Data is communicated from the backdoor to the KeyBoy command and control server using a simple encoding scheme. The encoder takes the data for exfiltration and multiplies each non-zero byte by `0x71 (113)`, followed immediately by evaluating this product modulo 256 (**see: Figure 1**).

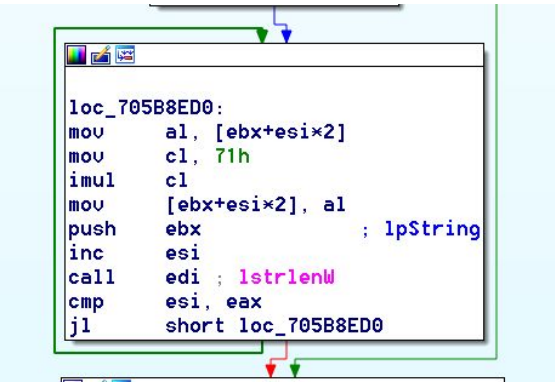

Figure 1: Typical encoding loop

During interaction with the backdoor, the malware operator can issue commands to direct the activities of the implant. These commands are received, decoded, parsed, and then evaluated to select the action to perform on the victim system. The method responsible for decoding the inbound commands from the C2 uses an identical function, however the decoding constant is `0xF7` instead of `0x71`.

While at first glance it may seem this encoding method is not well-defined or reversible once the data is collected by the operator, we show this is not the case. Upon closer inspection, it can be seen that this encoder is applying a fundamental concept from number theory in order to guarantee a mathematically well-defined data transmission function.

By way of example, consider the transmission of login and heartbeat data from the malware to the command and control server, as displayed in **Figure 2**.

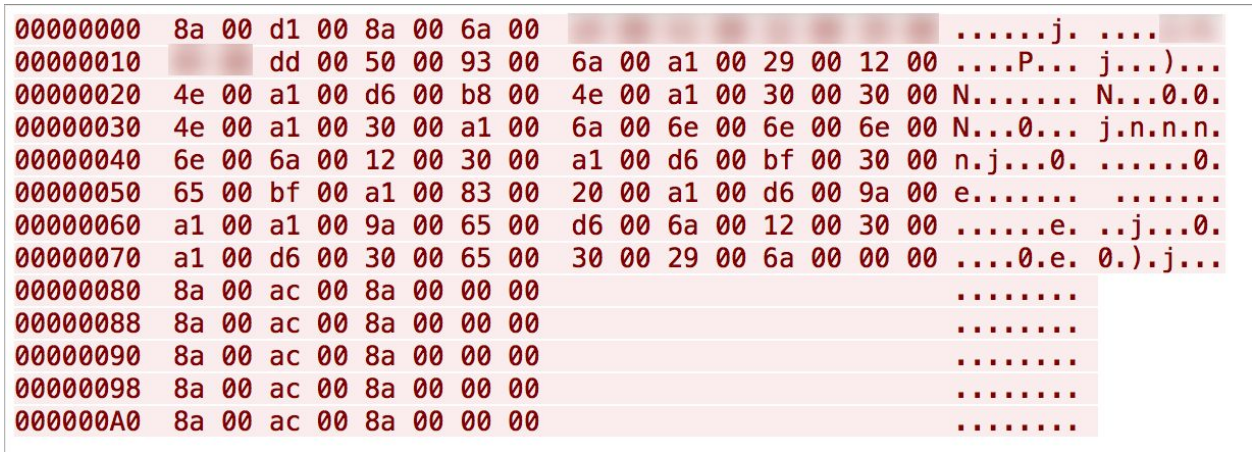

Figure 2: An example heartbeat beacon sent to the C2 server

Consider the initial byte from this beacon, hexadecimal `0x8A`. We know from analysis of the malware that this first byte is the result of encoding the * character, which has hexadecimal value `0x2A`. Thus the encoder performs the calculation $0x2A \times 0x71 = 0x128A$, as we know that outbound traffic from this malware sample uses static value 0x71 as the encoding constant.

$$0x2A \times 0x71 = 0x128A$$

or, in decimal notation:

$$42 \times 113 = 4746$$

Once this value is calculated, the algorithm takes the result modulo 256, leaving us with `0x8A` (decimal 138), the byte which is ultimately transmitted.

In order to successfully reverse this process and decode the transmitted byte, the command and control server relies on a simple concept which can be found in any number theory textbook:

1) If a, n are integers where $n > 0$, then a has a multiplicative inverse modulo n if and only if a and n are relatively prime.[1]

Or, alternately stated, that if a and n are relatively prime, there exists an a^{-1} such that:

$$a \cdot a^{-1} \equiv 1 \ (mod\ n)$$

Considering our sample case, because the number `0x71` (decimal 113) is prime, it follows that 256 and 113 must be relatively prime. Therefore the theorem outlined above implies that there must exist an integer a^{-1} (the *multiplicative inverse* of $113\ mod\ (256)$) such that

[1] Two integers are said to be *relatively prime* if the only common positive factor they share is 1.

$$a^{-1} \times 113 \equiv 1 \ (mod\ 256)$$

Solving for this integer a^{-1} yields the number 145. Therefore the command and control server needs only to perform the same encoding operation using this multiplicative inverse in order to decode the data received from the malware:

$$145 \times 138 \ (mod\ 256) \ = 42$$

Or, using hex notation:

$$0x91 \times 0x8A \ (mod\ 256) = 0x2A$$

Mathematically, this is a specific example of the general result, based on the above noted theorem. Thus for a positive integer k , if :

$$k \cdot a \equiv c \ (mod\ n)$$

then it follows that

$$k \cdot a \cdot a^{-1} \equiv c \cdot a^{-1} \ (mod\ n) \ \Rightarrow \ k \equiv c \cdot a^{-1} \ (mod\ n)$$

A Python script was created for the purpose of decoding the outbound communication from this backdoor.