Bad USB

written by Mert SARICA | 3 November 2014

Every year, in August, at the end of the traditional Black Hat Cybersecurity Conference held in Las Vegas, USA, two researchers, Karsten NOHL and Jakob LELL, signed a striking presentation called BadUSB.

In this presentation, in short, it was provided that the hardware software (firmware) used by the microcontroller in USB can work in a different way than expected by patching it. For this, the researchers first obtained the hardware software used by the microcontroller, then using Wireshark, they detected the commands used during the hardware software update. Then, within a period of less than 2 months, they analyzed the hardware software using reverse engineering, loaded their own commands into the unused areas of the original hardware software, and thus created a new hardware software, and loaded it onto the USB memory, and completed the process. After this, the USB memory connected to the target system could interact with the system by running the commands that the user had loaded into the memory with the hardware software.

What is the difference between BadUSB and other tools like Teensy or USB Rubber Ducky that also allow command execution when connected to the target system via USB? In practice, there is not much difference. In social engineering tests carried out with BadUSB, it may be possible that the chance of being caught/detected is somewhat lower both system-wise and visually compared to others. In terms of cost, you can see that it is possible to make BadUSB for cheaper than a 20$ Teensy or a 40$ Rubber Ducky.

If you visit the website of the researchers who work on BadUSB, you can see that they have not published the POC (proof-of-concept) codes for this study. There is no need for those who, like me, want to create a BadUSB to be disappointed because two security researchers, Adam Caudill and Brandon Wilson, also made a similar study and presented it at the DerbyCon Information Security Conference at the end of September, and also uploaded the tools they developed during the research to GitHub along with the source codes.

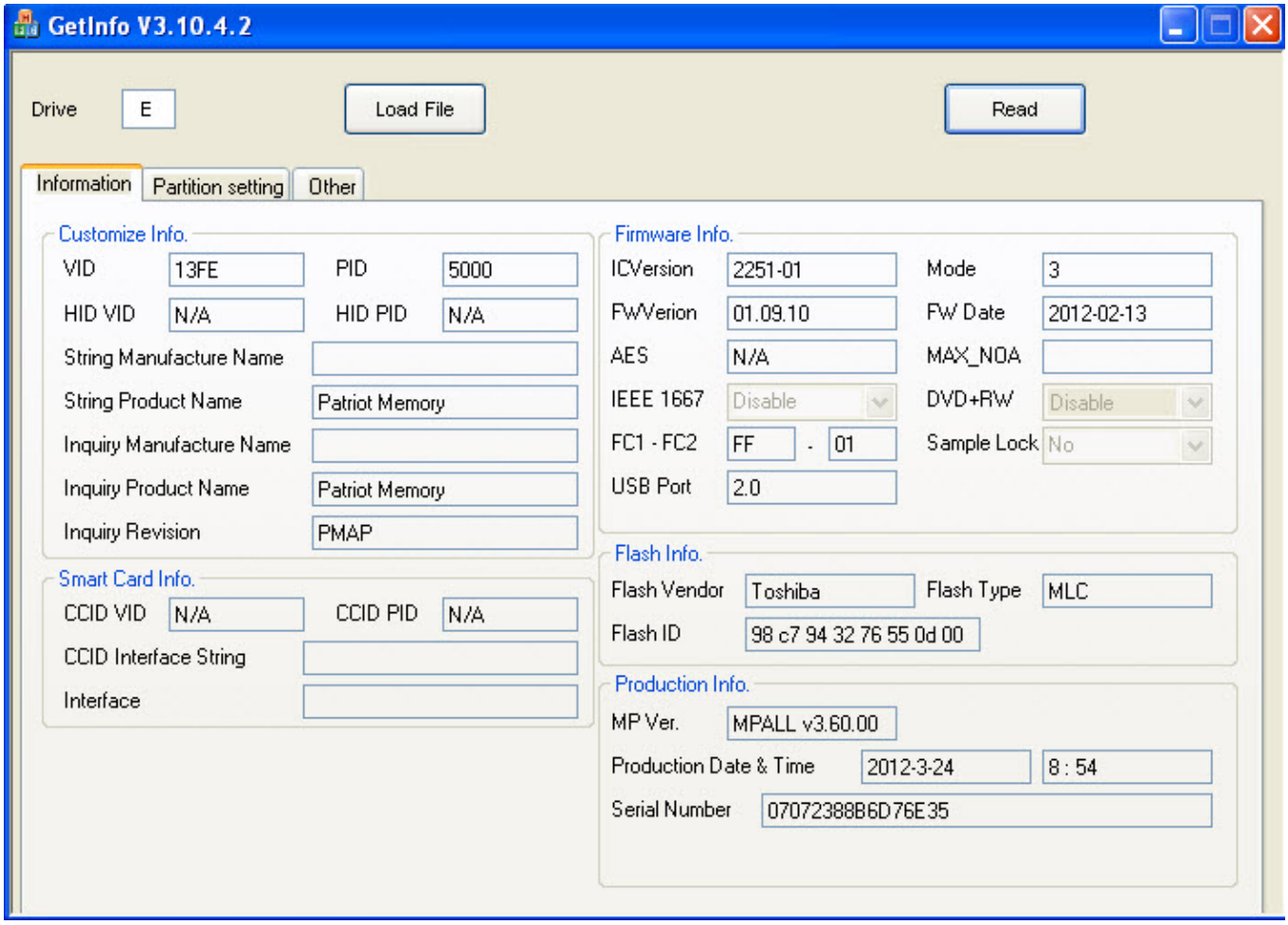
After looking at the presentation file and codes, I also got to work creating a BadUSB. I ordered a Patriot Xpress model USB memory with a Phison microcontroller, which researchers use, from Amazon because it was not

available in Turkey.

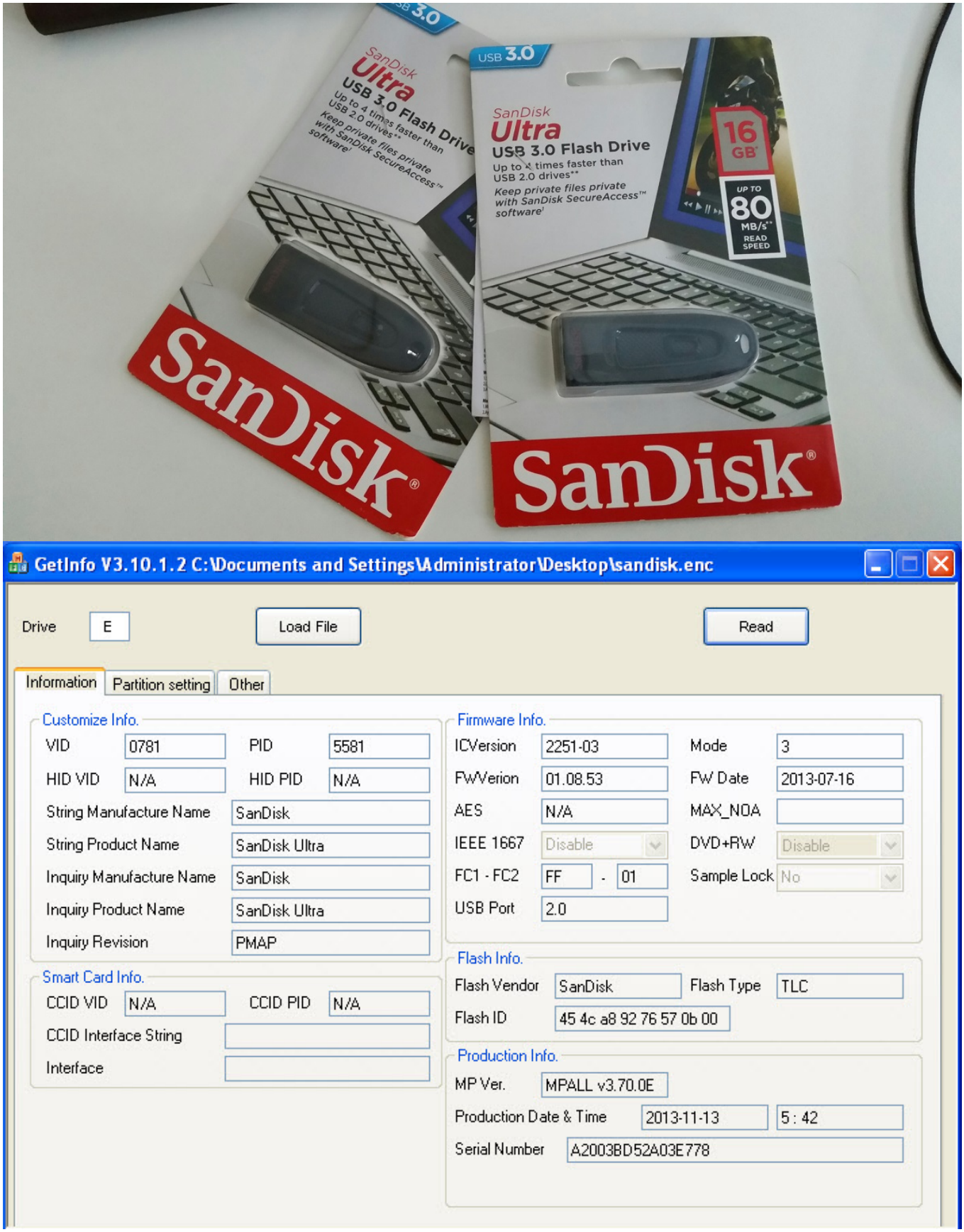
If you look at the presentation file, you will see that researchers are working on the Phison PS2251-03 model, so the USB memory that will be used for the tool they developed must have this model microcontroller in order for it to function.

After the USB memory arrived, I checked the model using the GetInfo tool (or you can use Chip Easy tool as well), and I was disappointed to find that the model was different. So this time, I set out to hunt for a Phison brand PS2251-03 model USB memory.



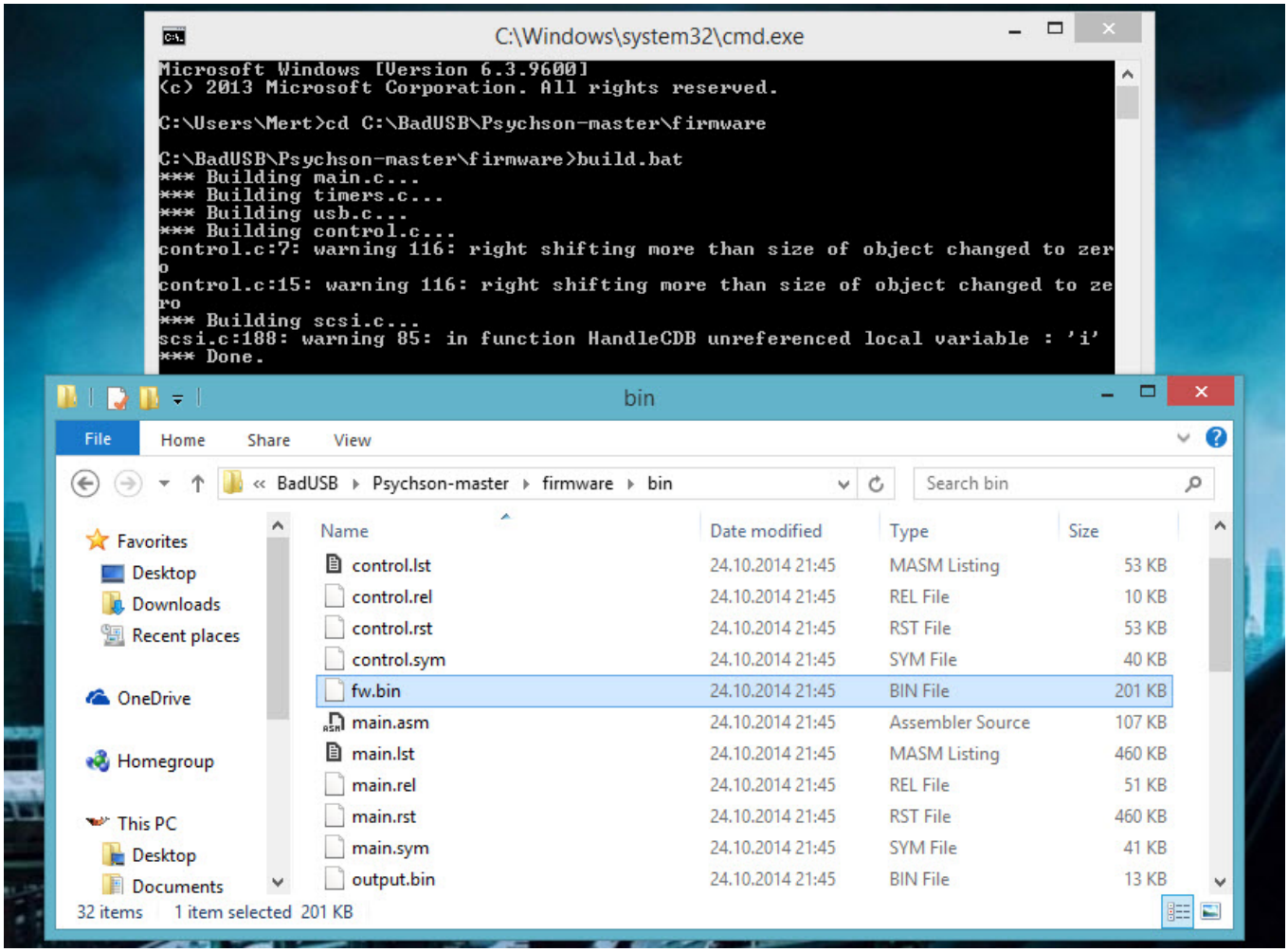


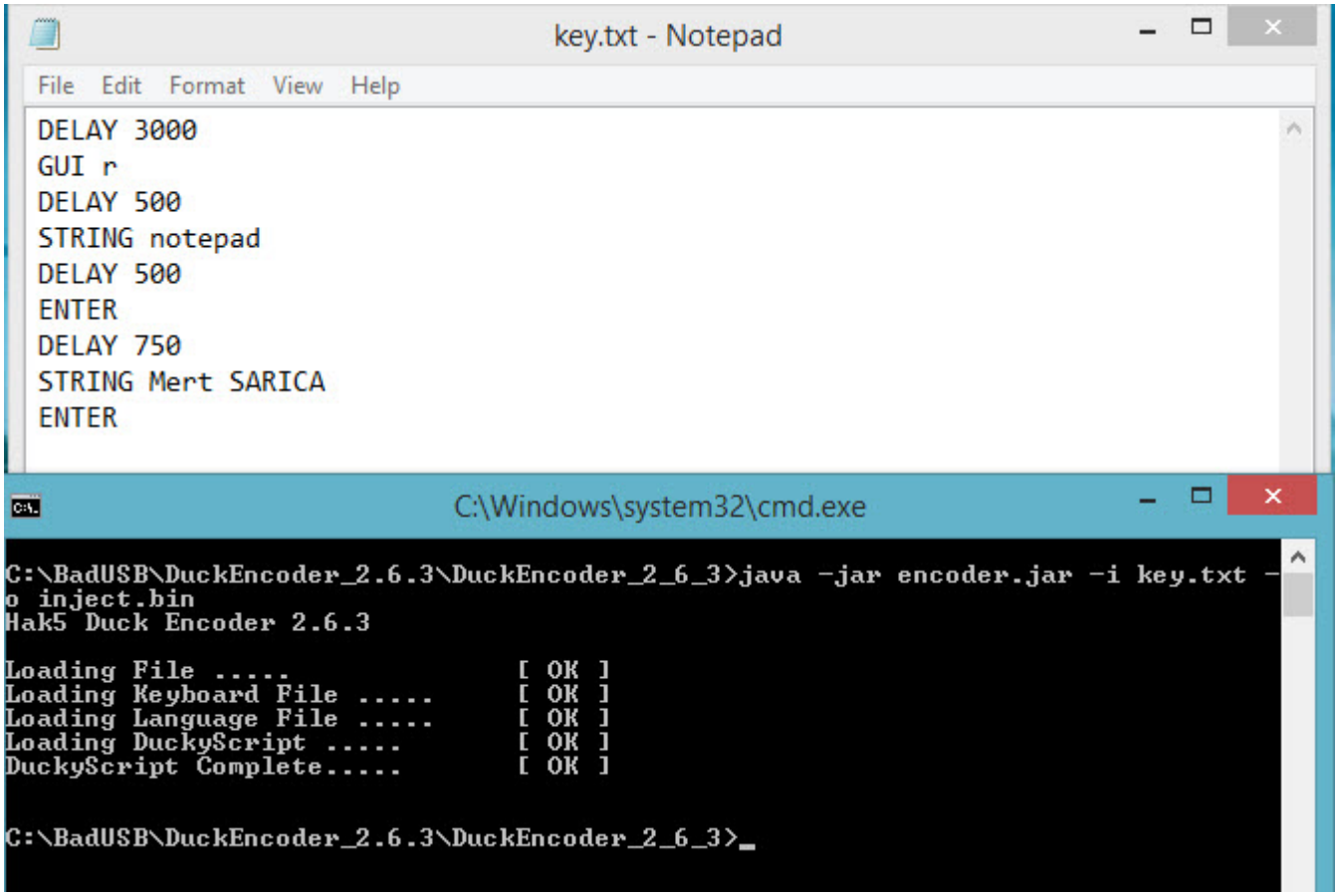
Like me, many users around the world set out to hunt for USB memories and provide feedback to researchers, so the researchers decided to compile a list of USB memories that have the potential to be BadUSBs. While checking this list from time to time, I decided to buy Sandisk Ultra 16 GB USB memory (SDCZ48-016G-U46) that I saw by chance while browsing at Teknosa (24 TL) and check its model. With great enthusiasm, I opened the package and, after checking it with the GetInfo tool, I saw that it was Phison’s model, the supported model PS2251-03, so I proceed with the BadUSB creation steps on the GitHub page.



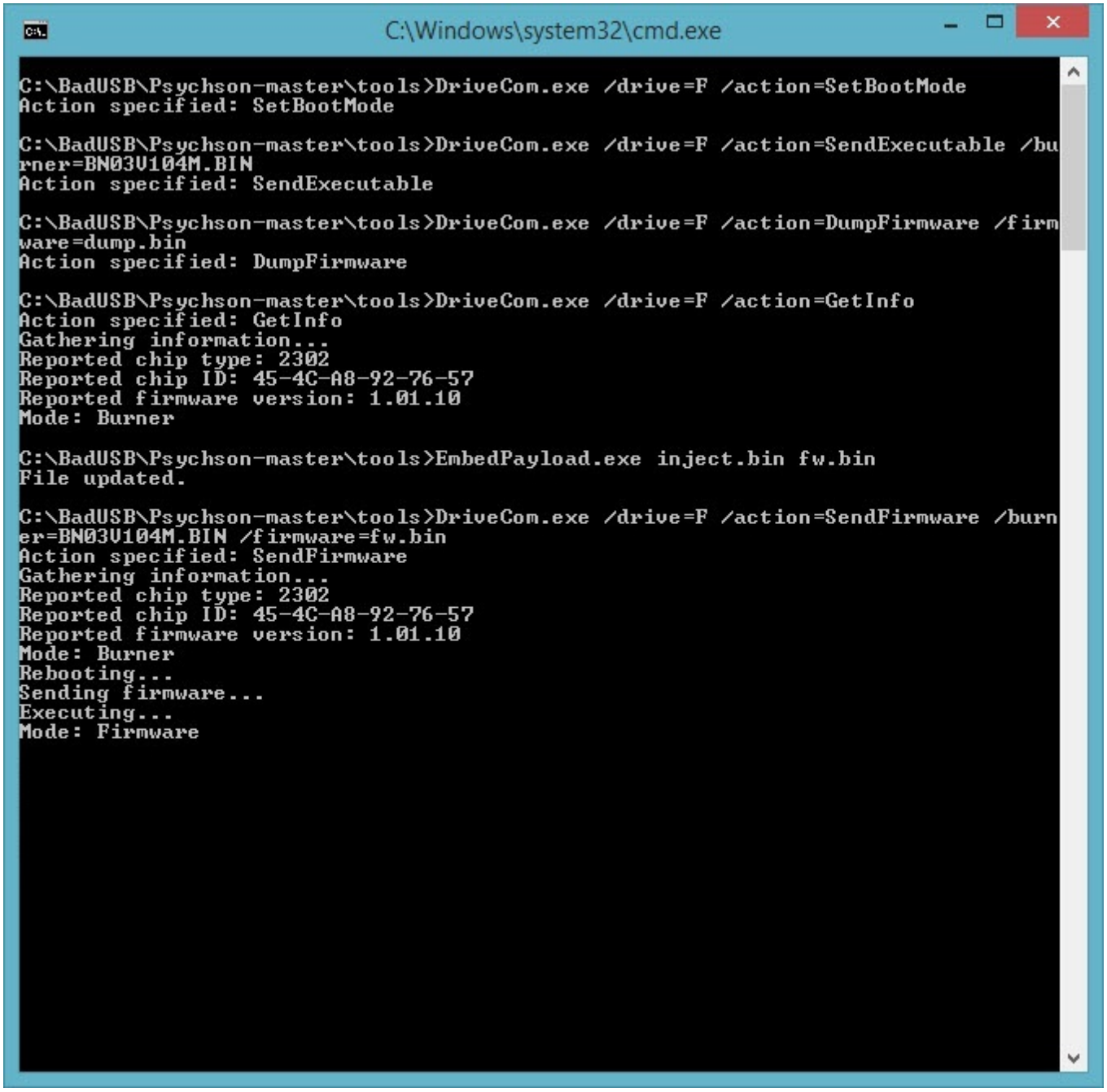
Unfortunately due to a mistake in one of the steps, I had to throw away this disk I bought. Afterwards, I bought two more Sandisk Ultra, and again due to a mistake, I had to throw away one of the disks. I decided to proceed carefully with the BadUSB creation steps. After compiling the hardware software, when it came to creating a set of commands in Ruby Ducky format, I

used Duckencoder tool to create a simple command set of run -> notepad -> Mert SARICA. (I followed the instructions in the Running Demo 1 (HID Payload) section in the ReadMe file.)





Finally, after successfully passing all the steps below, I was able to create a BadUSB :) Unfortunately, with Sandisk's Ultra model, there is only one chance to update the hardware software, so it's a one-shot deal for now, but work is ongoing in this area, so it may be useful to have one of these brand models USB on hand for use in social engineering tests.



If you were to ask what measures can be taken against BadUSBs as an organization, you can ban the use of USBs across the organization. If this is not possible, you can choose to use products like IronKey, which have signature control against hardware software updates, across the organization. Additionally, training employees on identifying suspicious USBs and not to use unknown or non-authorized USBs can also help in preventing badUSB attacks.

Hope to see you in the following articles.

Note: You can watch the video I prepared about the USB that has been converted to BadUSB from the link below.