УДК 338.22:004
DOI 10.18101/978-5-9793-1497-6-76-81

# ИЗУЧЕНИЕ ПРОТОКОЛА USB И МЕТОДОВ ФОРЕНЗИКИ НА ПРИМЕРЕ РЕШЕНИЯ ОДНОГО ИЗ ЗАДАНИЙ С СОРЕВНОВАНИЙ SARCTF

**© Конькова Анна Евгеньевна**
студент,
Бурятский государственный университет имени Доржи Банзарова
Россия, 670000, г. Улан-Удэ, ул. Смолина, 24а
E-mail: konkova@yandex.ru

**© Немчинова Татьяна Владимировна**
кандидат педагогических наук, доцент,
Бурятский государственный университет имени Доржи Банзарова
Россия, 670000, г. Улан-Удэ, ул. Смолина, 24а
E-mail: ntv05@mail.ru

CTF – соревнования по спортивному хакингу или командная игра, главной целью которой является захват «флага» у соперника. Формат Capture the flag может использоваться как в пейнтболе, так и среди ролевиков, и в компьютерных играх. В области компьютерной безопасности есть свое понятие CTF. Есть два типа соревнований. Первый – Attack-Defence – считается классическим, поскольку использует правила Capture the Flag в чистом виде. Основная идея CTF — искать уязвимости в своей системе и атаковать чужие. В статье хочется поделиться опытом изучения протокола USB и методов форензики на примере решения одного из заданий этого соревнования.
**Ключевые слова:** протокол, дамп, форензика, соревнование, компьютерная криминалистика, пакет, USB.

В соревнованиях формата CTF (на русский переводится как «захват флага») есть категория заданий по форензике. Форензика, или компьютерная криминалистика изучает инциденты, связанные с компьютерной информацией. В этой категории часто попадаются задания на изучение дампов сетевых пакетов. Это может быть интернет-, блютуз-, USB-траффик и т.п, не важно, алгоритм примерно один.

Во время решения одного из заданий SarCtf 2020 мне попался файл под названием usb_here.pcapng. По расширению pcapng можно предположить, что это дамп траффика USB, полученный с помощью сниффера. Организаторы CTF любят путать участников, маскируя файлы всевозможными способами: меняя расширение, пряча один в другом и так далее.

Но если открыть файл в программе Wireshark (рис. 1), становится ясно, что это действительно дамп с USB.

## Немного о протоколе USB

Первые пакеты – это пакеты-маркеры, которые настраивают соединение между USB-устройством и компьютером. Посмотрим пример SETUP-запроса:

**bmRequestType** – битовое поле, которое содержит характеристики запроса (Рис. 2),

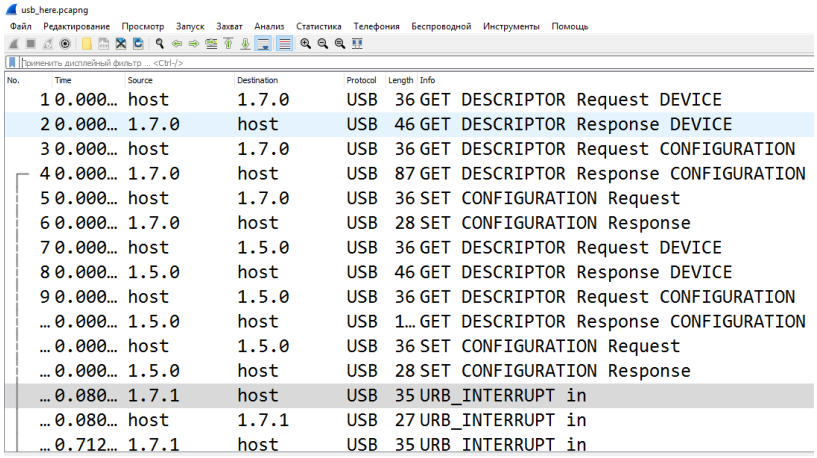


Рис. 1.

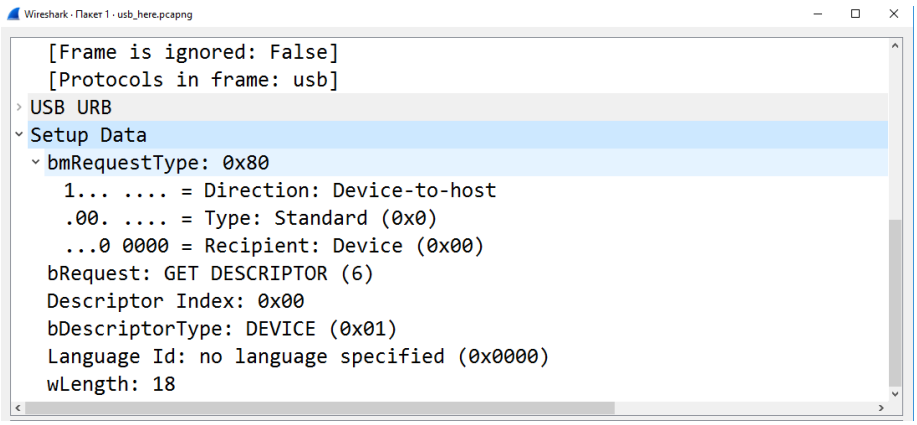


Рис. 2.

Бит 7: Направление передачи данных в DATA фазе:

- 0 = от хоста к устройству
- 1 = от устройства к хосту

В поле Direction единица, значит запрос направлялся от устройства к хосту.
Биты 6...5: Тип запроса

- 0 = стандартный запрос USB
- 1 = стандартный запрос для определенного класса устройств USB
- 2 = пользовательский запрос
- 3 = зарезервировано

Type: 0. Значит, это стандартный запрос USB.
Биты 4...0: Куда адресован запрос

- 0 = устройству
- 1 = интерфейсу
- 2 = конечной точке
- 3 = другое

- 4...31 = зарезервировано

**bRequest** – уникальный код запроса.

Здесь: 6 = GET_DESCRIPTOR

Далее идет поле Descriptor.

Descriptor-type: Devise

Дескриптор устройства всегда один, он содержит базовую информацию об устройстве (код производителя, код устройства, класс устройства и т. д.)

Далее идут пакеты in, отвечающие за передачу данных. Один пакет – от устройства к хосту – передает данные, второй – от хоста к устройству - подтверждает получение с помощью флага USBD_STATUS_SUCCESS (Рис. 3).

Теперь необходимо узнать тип устройства, с которого был снят дамп. Для этого нужно найти пакет GET DESCRIPTOR RESPONSE, а в нем строку idproduct. И потом с помощью поисковика определить тип устройства (мышь, клавиатура, или устройство памяти). В нашем случае это клавиатура.

После ищем поле “Leftover Capture Data”. Собственно, это и есть передаваемые устройством данные.

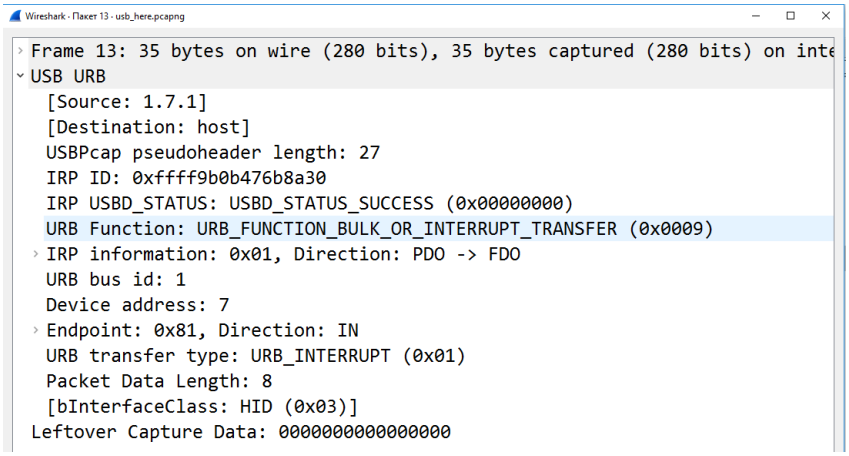


Рис. 3

Пункт “Leftover Capture Data” представляет собой строчку длиной 8 байт. Код кнопки, нажатой на клавиатуре, находится в 3-м байте.

Пакеты с полезными данными имеют длину 35, зная это, можно их отсортировать. Просто кликнув по цифре 35 в любом из пакетов, далее - Применить, как фильтр -> Выбрано (рис. 4).

Теперь отсталось извлечь и расшифровать данные. Для этого кликнем по полю “Leftover Capture Data” -> Применить, как Столбец, а все остальные колонки наоборот скрываем. Выделяем то, что осталось и скопируем их в текстовый документ cap.txt.

Первая функция – capSort. “Leftover Capture Data” это множество восьмибайтных строк, но из всей этой информации нам нужен только 3-й байт каждой строки. После обработки cap.txt с помощью capSort, все третьи байты будут записаны в список keyMass.

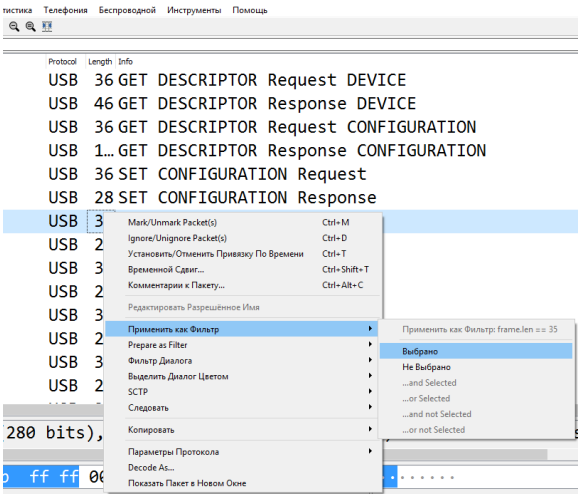


Рис. 4

Функция capRead расшифровывает keyMass. Работает это так: в словарь codes записать скан-коды для USB-клавиатуры. Далее каждый байт преобразуется в целое десятичное число с помощью команды key=int(i, 16). Можно было бы оставить коды шестнадцатеричными, но лично удобнее всего видеть цифры в десятичном представлении. После этого ищем номер клавиши в codes. Если клавиша нашлась, записываем ее в строку decrypted. Неизвестные клавиши обозначим звездочкой. Результат работы скрипта, можно видеть на рис. 6:

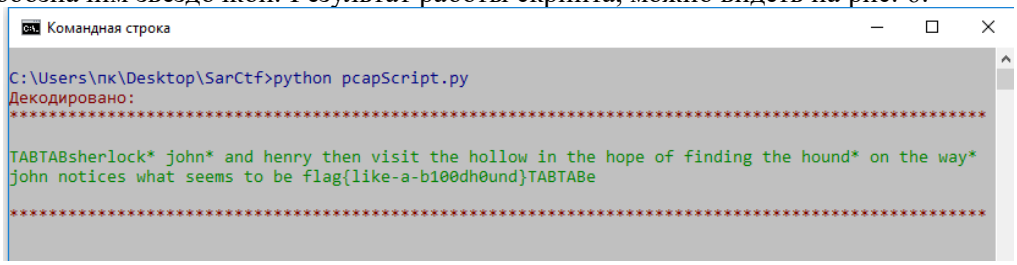


Рис. 6. Результат работы скрипта

TABTABsherlock* john* and henry then visit the hollow in the hope of finding the hound* on the way* john notices what seems to be flag{like-a-b100dh0und}TABTABe

А вот и флаг: flag{like-a-b100dh0und}.

Примечание: такой скрипт обрабатывает только одиночное нажатие клавиш, то есть, он не заметит сочетания вроде “SHIFT+буква”, но, как видно, здесь таких и нет.

Таким образом, знание протоколов передачи данных может пригодиться в самых разных областях, начиная с системного администрирования и заканчивая

форензикой. На примере задания с CTF, показано, что, имея лишь дамп, можно достоверно восстановить переданные/полученные данные: это могут быть посещенные веб-страницы, отосланные/загруженные по блютуз файлы, текст, введенный с клавиатуры или даже движения мыши. Все это может быть использовано, как во благо, так и во вред.

**Код программы**

```
import colorama
from colorama import Fore
colorama.init()
keyMass=[]
def capSort(): # Считывает 3-й байт каждой строки
   file=open('cap.txt','r')
   for x in file:
      s=x[4:6]
      keyMass.append(s)
   file.close()
   return keyMass
def capRead(): # Расшифровка
   decrypted=""
   codes={2:"",4:"a",5:"b",6:"c",7:"d",8:"e",9:"f", 10:"g", 11:"h", 12:"i", 13:"j",
14:"k", 15:"l", 16:"m", 17:"n", 18:"o", 19:"p", 20:"q", 21:"r", 22:"s", 23:"t", 24:"u",
25:"v", 26:"w", 27:"x", 28:"y", 29:"z", 30:"1", 31:"2", 32:"3", 33:"4", 34:"5", 35:"6",
36:"7", 37:"8", 38:"9", 39:"0", 40:"ENTER", 41:"ESC", 42:"\b", 43:"TAB", 44:" ",
45:"-",47:"{", 48:"}", 56:"/", 57:"CAPS", 79:"RIGHTARROW",
80:"LEFTAAROW",190:".",188:","}
   for i in keyMass:
      key = int(i, 16)
      if key!=0:
         if key in codes:
            decrypted=decrypted+codes[key]  # Если клавиша нашлась,
добавить в строку decrypted
         else:
            decrypted=decrypted+"* "
   print(Fore.RED + "Декодировано:")
   print(Fore.RED + '*'*100)
   print(Fore.GREEN + decrypted)
   print()
   print(Fore.RED + '*'*100)
capSort()
capRead()
```

**Литература**

1. Andries Brouwer. Keyboard scancodes. [Электронный ресурс]. URL: https://www.win.tue.nl/~aeb/linux/kbd/scancodes-14.html.
2. CTF Wiki. USB. [Электронный ресурс]. URL: https://ctf-wiki.github.io/ctf-wiki/misc/traffic/protocols/USB.

3. Интерфейс USB. Завершение реализации. Сообщество EasyElectronucs.ru. [Электронный ресурс]. URL: http://we.easyelectronics.ru/electro-and-pc/interfeys-usb-zavershenie-realizacii.html.

## STUDYING THE USB PROTOCOL AND FORENSIC METHODS ON THE EXAMPLE OF SOLVING ONE OF THE TASKS FROM THE SARCTF COMPETITIONS

*Anna E. Konkova*
Student,
Dorzhi Banzarov Buryat State University
24a Smolina St., Ulan-Ude 670000, Russia
E-mail: konkova@yandex.ru

*Tatiana V. Nemchinova*
Cand. Sci. (Education), A/Prof.,
Dorzhi Banzarov Buryat State University
24a Smolina St., Ulan-Ude 670000, Russia
E-mail: ntv05@mail.ru

CTF - sports hacking competition or team game, the main purpose of which is to capture the "flag" from the opponent. The Capture the flag format can be used both in paintball and among role-playing players, and in computer games. The field of computer security has its own concept of CTF. There are two types of competitions. The first - Attack-Defense - is considered as classic because it uses the Capture the Flag rules in its purest form. The main idea of CTF is to look for vulnerability in its system and attack the other ones. The authors of the article would like to share the experience of studying the USB protocol and forensic methods on the example of solving one of the tasks of this competition.
*Keywords*: protocol, dump, forensics, competition, computer forensics, package, USB.