Rekin w strumieniu...

...czyli hakujemy Stream Decka

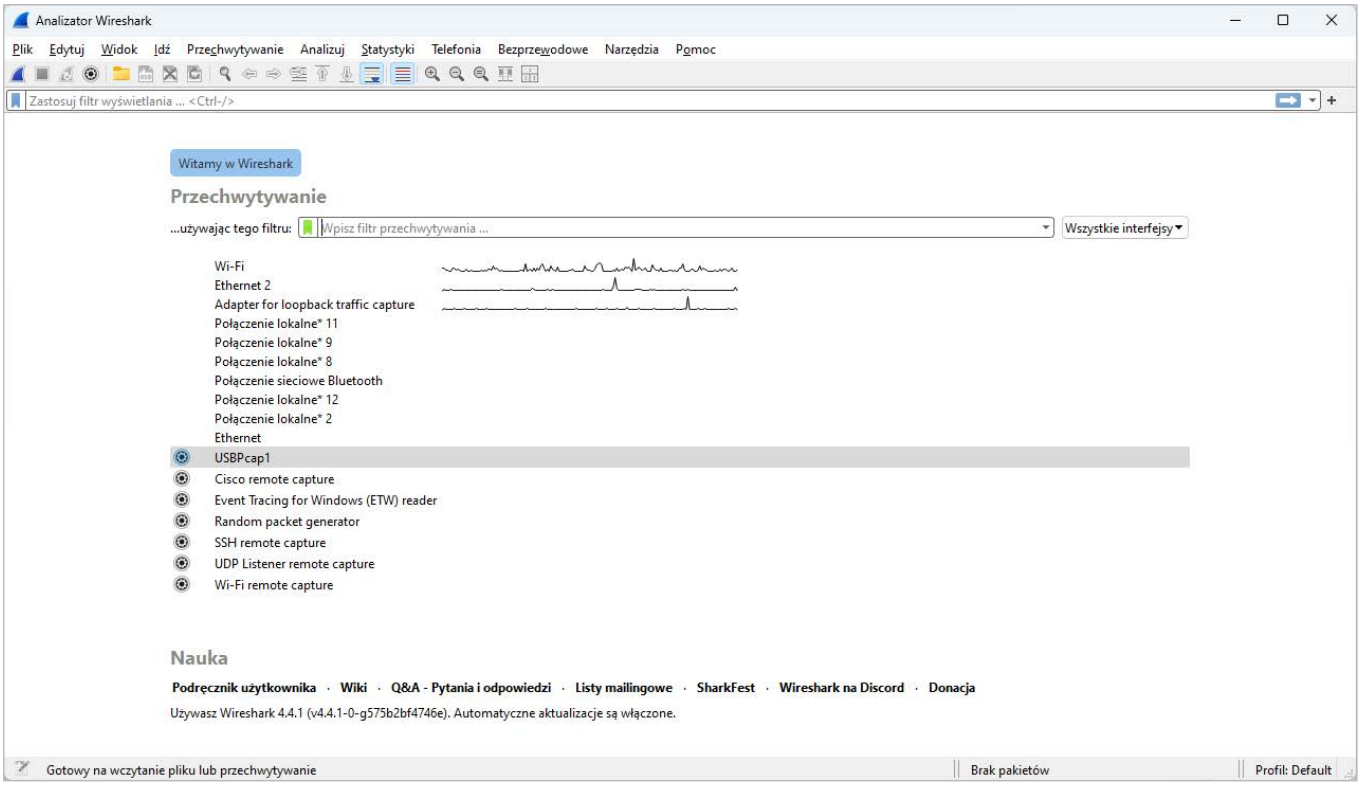
Jak powszechnie wiadomo, jestem wielkim fanem koncepcji klawiatur makro: małych urządzeń, które pomagają automatyzować często powtarzane zadania. Jednym z pierwszych komercyjnych, zaawansowanych urządzeń tego typu jest Stream Deck zaprojektowany przez firmę Elgato, przejętą jakiś czas temu przez Logitecha. Wyposażony jest on w przezroczyste przyciski zamontowane ponad wyświetlaczem LCD, co pozwala każdemu z nich dynamicznie przyporządkowywać inne ikony.

Razem ze Stream Deckiem otrzymujemy również od Elgato odpowiednie oprogramowanie. Pozwala ono przyporządkowywać przyciskom akcje, organizować je w ekrany, te w foldery, a całe zestawy konfiguracji możemy również zamykać w profilach, które aktywowane są, gdy określona aplikacja przejdzie na pierwszy plan. Jeśli natomiast chodzi o możliwe do wywołania akcje, to razem z aplikacją otrzymujemy dosyć pokaźny ich zbiór, który w razie potrzeby możemy jeszcze rozszerzać za pomocą pobieranych z internetowego sklepu wtyczek.

W większości przypadków oprogramowanie dostarczone przez producenta jest w zupełności wystarczające. Istnieją jednak takie sytuacje, w których korzystanie z tej pożytecznej klawiatury może okazać się znacząco utrudnione. Być może pracujemy w środowisku, gdzie dostępne jest wprawdzie środowisko programistyczne, ale instalowanie programów jest znacząco utrudnione lub wręcz niemożliwe? Albo też chcielibyśmy sterować pisanym przez siebie programem bezpośrednio, a nie za pośrednictwem zewnętrznego oprogramowania? Albo być może mamy zupełnie inny pomysł na wykorzystanie urządzenia z programowalnymi przyciskami? Lub – który to powód jest równie dobry jak wszystkie inne – chcemy po prostu dowiedzieć się, jak odbywa się komunikacja Stream Decka z oprogramowaniem na komputerze? W takich sytuacjach konieczne może okazać się podjęcie nieco bardziej… niekonwencjonalnych działań.

Naszą hakerską podróż odbędziemy w trzech etapach. Na początku konieczne będzie pozyskanie próbki komunikacji Stream Decka z dedykowanym oprogramowaniem. W drugim kroku przeprowadzimy analizę tejże próbki i spróbujemy rozszyfrować, w jaki sposób kodowane są przesyłane w obie strony dane. Na koniec spróbujemy napisać małą demonstracyjną aplikację, która nawiąże komunikację ze Stream Deckiem, wyświetli zdefiniowane ikony pod przyciskami oraz zareaguje na ich wciśnięcia.

Zabierzmy się więc do roboty.



Rysunek 1. Uruchamiamy Wiresharka

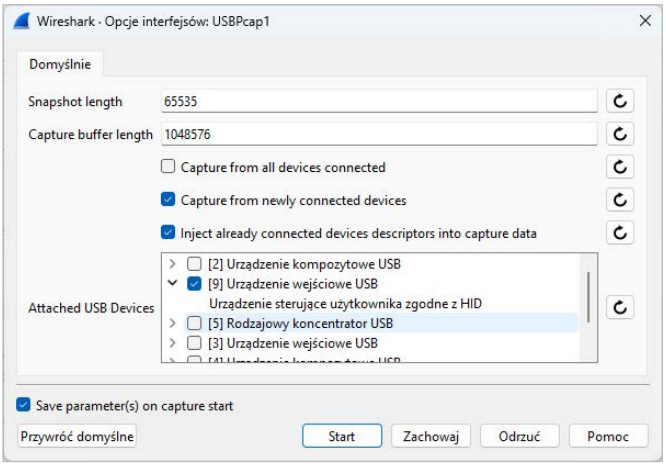
WIRESHARK

Stream Deck wymienia z komputerem informacje poprzez port USB, a transmisję taką można podsłuchać. Co więcej, aby to zrobić, nie musimy korzystać z żadnych urządzeń – wystarczy tylko zastosować odpowiednie oprogramowanie.

Bardzo popularną aplikacją, która pozwala wykonywać zrzuty komunikacji sieciowej, jest Wireshark. Okazuje się jednak, że oprócz wychwytywania pakietów sieciowych jest ona również w stanie kontrolować komunikację odbywającą się z urządzeniami poprzez porty USB. Aby stało się to jednak możliwe, podczas instalacji Wiresharka musimy zadbać o zaznaczenie pola wyboru przy pozycji „USBPcap".

Wireshark wraz z USBPcap to jedyne, co będzie nam na ten moment potrzebne. Uruchamiamy więc ten analizator i z listy dostępnych źródeł przechwytywania wybieramy interfejs USBPcap (Rysunek 1).

Zanim przejdziemy do przechwytywania transmisji, warto jest najpierw ograniczyć ten proces tylko do pojedynczego urządzenia, które nas interesuje. Oszczędzi nam potem wyławiania interesujących nas informacji z szumu. W tym celu klikamy małą ikonkę koła zębatego obok nazwy USBPcap, a następnie odszukujemy Stream Decka na liście (Rysunek 2).



Rysunek 2. Konfigurujemy USBPcap

Odnalezienie odpowiedniego urządzenia może zająć chwilę czasu (być może metodą prób i błędów), ponieważ nie znajdziemy tu bezpośrednio jego nazwy. Stream Deck jest jednak urządzeniem wejściowym typu HID (ang. *Human Interface Device*), co już na starcie zawęża listę „podejrzanych". W razie kłopotów możemy też tymczasowo odłączyć część urządzeń, aby szybciej znaleźć to, które nas interesuje. W moim przypadku Stream Deck dostępny był jako urządzenie wejściowe USB o indeksie 9.

Teraz możemy uruchomić przechwytywanie, klikając podwójnie nazwę USBPcap.

Bardzo ważnym jest, aby wymusić na urządzeniu i towarzyszącym mu oprogramowaniu przeprowadzenie tych rodzajów transmisji, na których najbardziej nam zależy. Ponieważ chcemy przede wszystkim móc samodzielnie wyświetlać na Stream Decku własne ikony oraz przechwytywać wciśnięcia przycisków, to właśnie tego typu transmisje musimy teraz sprowokować. Wysłanie informacji o wciśnięciu przycisku jest oczywiście bardzo łatwe, natomiast ten pierwszy rodzaj transmisji wymusimy, gdy na przykład zmienimy aktywny ekran (choćby wchodząc do wirtualnego folderu – Rysunek 3).

Rysunek 3. Wyświetlamy osobny ekran na Stream Decku podczas przechwytywania transmisji

Po wykonaniu wszystkich operacji wciskamy drugi przycisk na pasku narzędzi, a przechwytywanie zostanie zakończone. Teraz czeka nas nieco detektywistycznej roboty.

ANALIZA PAKIETÓW

Przyjrzyjmy się teraz transmisji, która odbyła się pomiędzy Stream Deckiem a dedykowaną aplikacją na komputerze.

Stream Deck prowadzi komunikację w standardzie HID. Standard ten zaprojektowany został dla wszystkich urządzeń, które ze swojej natury stanowią rodzaj interfejsu pomiędzy komputerem a człowiekiem. Są to więc na przykład klawiatury, myszki, trackballe, touchpady, joysticki, kierownice i inne kontrolery gier i tak dalej. Znajomość tego standardu [1] oczywiście pomaga podczas analizy transmisji, ale w naszym przypadku – co zaraz się okaże – nie będzie wcale konieczna.

Skupmy się najpierw na przyciskach. Uruchamiamy więc przechwytywanie, a następnie wciskamy wszystkie przyciski na Stream Decku – najlepiej w jakiejś konkretnej kolejności, na przykład wierszami od lewego, górnego do prawego, dolnego.

Szybko zauważymy, że wciśnięcie każdego z przycisków generuje dwa zdarzenia oznaczone w Wiresharku jako „URB_INTERRUPT in". Kiedy wybierzemy jedno z takich zdarzeń, w dolnej części ekranu pojawi się podgląd informacji na jego temat (po lewej stronie) oraz szesnastkowy zrzut surowych danych (po prawej) – jak na Rysunku 4.

Pierwsze bajty szesnastkowego zrzutu stanowi tak zwany pseudonagłówek USBPcap. Narzędzie to dodaje do każdego zrzutu garść metadanych, które pomagają potem przeprowadzić bardziej szczegółową analizę. Oprócz samej transmisji USBPcap zapisuje między innymi adres urządzenia, kierunek transmisji, jej rodzaj oraz status.

Wireshark jest na szczęście skonstruowany w taki sposób, że pomaga szybko rozróżnić metadane dodane do pakietu przez USBPcap oraz surowe informacje przesłane tym konkretnym pakietem. Wystarczy kliknąć na bajty w zrzucie, aby po lewej stronie podświetlone zostało konkretne pole pseudonagłówka. W ten sam sposób możemy odnaleźć również surową transmisję (widać ją na Rysunku 4).

Rozszyfrowanie sposobu kodowania informacji o wciśnięciach klawiszy jest zadaniem wyjątkowo łatwym. Podczas analizy kolejnych pakietów szybko zauważymy, że każdy z nich zaczyna się od czterobajtowego nagłówka (01 00 0f 00). Kolejne 15 bajtów reprezentuje zaś

WORDPRESS JAKO HEADLESS CMS. INTEGRACJA Z NOWOCZESNYM FRONT-ENDEM DLA ELASTYCZNYCH APLIKACJI

INDEX: 285358

www.programistamag.pl

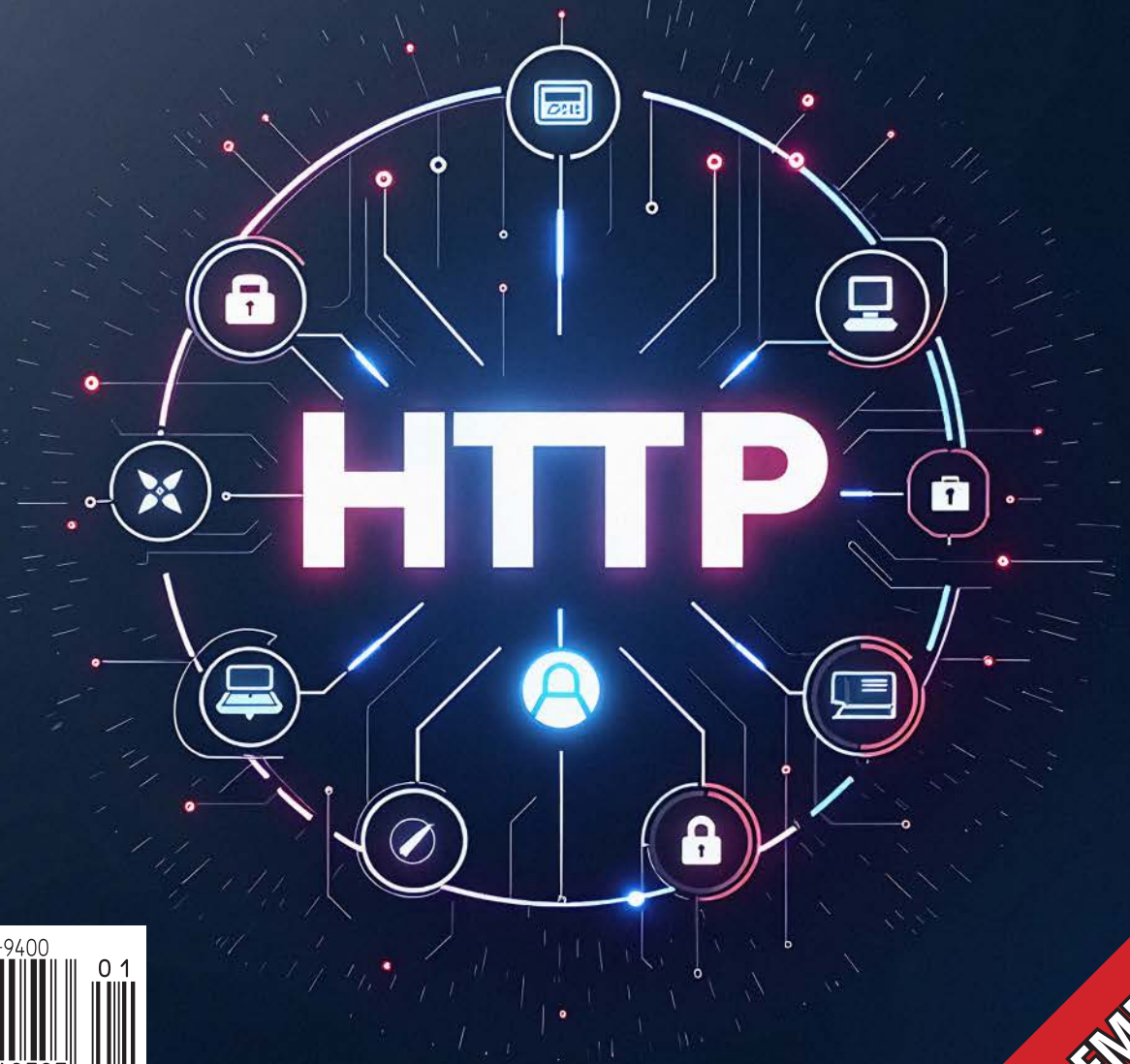
Magazyn programistów i liderów zespołów IT

programista

1/2025 (116)

Cena 32.90 zł (w tym VAT 8%)

JAK DZIAŁA INTERNET



ISSN 2084-9400

01

9 772084 940503

Współbieżność CSP w języku Go

Rekin w strumieniu... ...czyli hakujemy Stream Decka

Gwiazda m… …orutyny, czyli Sea… …aktyce

CSS… …lasses …adzie TailwindCSS

NOWY NUMER JUZ W EMPIKACH