UsoroSync: Praktyczna I teoretyczna dokumentacja protokołu transferu danych

Mateusz Dukat

Spis treści:

[TODO]
- zrobienie ~~generatora klucza prywatnego~~ I ogarnięcie rozesłania go na wszystkie urządzenia które chcemy.
- wprowadzenie ID urządzeń (im niższe ID, tym urządzenie ma być wyżej w stacku urządzeń == ma się połączyć pierwsze) co pozwoli zarządzać powiązanymi urządzeniami

1. Rozpoczęcie połączenia

1.0 Pre-connection

Każdy użytkownik posiada aplikacje (daemon, serwis) [nazywana później aplikacją] która jest usługą peer2peer. Na każdym urządzeniu znajduje się w/w usługa która jest zbudowana pod daną architekturę lub dany system.

- Wymagania poprawnego działania aplikacji:
  - plik konfiguracyjny przechowujący ustawienia
  - ***klucz prywatny*** (identyczny na różnych urządzeniach)
  - plik binarny aplikacji

Na chwilę obecną kluczem prywatnym będzie losowy, 256bitowy (32bajty) ciąg znaków w zakresie znaków ASCII: 0x21 – 0x7E (DEC 33 – 126).

1.1 Otwarcie socketu, handshake

Aplikacja rozpoczyna połączenie od otwarcia socketu na porcie 1488 w typie TCP/IP. Socket ma budowe peer'a (lub node'a) I może zarazem odbierać jak I wysyłać dane od drugiego, zbudowanego tak samo Socketu.

Dla ograniczenia wielkości pakietów podczas transferu, polecenia (GET, PUT, ACK, itd) są rozsyłane w postaci jednego bajtu (8 bitów). [Patrz tabela komend].
Wielkość buffora transmisji danych uznajemy za 256 bajtów, resztą zajmuje się biblioteka Socket dla danego urządzenia (dzielenia bufferu na mniejsze części I transfer warstwą TCP).

Handshake polega na wysłaniu drugiej stronie komendy "HDS" z hashem md5 klucza prywatnego zasolonego kluczem hardkodowanym, gdzie w losowych miejscach zostają wstawione losowe bajty **(z zakresu DEC 48-57 I 97-102)** polegające na "próbie klucza". Druga strona musi sprawdzić hash md5 z solą swojego klucza I sprawdzić które bajty zostały wstawione, wtedy z tych bajtów wykonać kolejny hash md5 z solą klucza hardkodowanego I odesłać go w odpowiedzi.

Przykład w/w algorytmu został zaprezentowany w pliku "algo1.py" w katalogu "/Usoro/python/testy/".

Strona o niższym ID:

Losowe_bajty[4] = rand(), rand(), rand(), rand()
Hash_md5 = md5(Klucz Prywatny, Klucz Hardkodowany)
wstaw losowo Losowe_bajty do Hash_md5
wyślij Hash_md5 z losowymi bajtami do drugiej strony
Pakiet = Komenda HDS, sizeof(Hash_md5), Hash_md5, 0x00*

**sizeof(Hash_md5) ma wielkość typu fixed o wielkości 1 bajta*

Strona o wyższym ID:

Hash_md5 = md5(Klucz Prywatny, Klucz Hardkodowany)
rozpoznanie które bajty się nie zgadzają
Powrotna_wiadomość = md5(Bajty z hashu, Klucz Hardkodowany)
odesłanie pakietu Powrotna_wiadomość

Strona o niższym ID:

sprawdź czy hashe bajtów się zgadzają
md5(Losowe_bajty, Klucz Hardkodowany) == Powrotna_wiadomość
jeśli się zgadzają, kontynuacja połączenia
jeśli nie, przerwij połączenie niezwocznie

Obecnie za hardkodowany klucz uznajemy ciąg bajtów “2137” który w postaci binarnej wygląda następująco:

0x02, 0x01, 0x03, 0x07

Dla większego bezpieczeństwa możemy w kodzie programu wykonać działanie którego sumą będzie ciąg 2137, np:

```
int hardcode[4];
hardcode[0] = 4/2;
hardcode[1] = (997-996.5)*2;
hardcode[2] = sqrt(9);
hardcode[3] = 6++;
```

Spowoduje to brak ciągu znaków “2137” w sekcji .data kodu, co wpłynie na czas próby inżynierii odwrotnej pliku binarnego I odnalezienia klucza hardkodowanego.

2. Czas łączności

2.1 Sprawdzenie czy pliki w folderach się zgadzają

MD5 jest zajebiście fajną funkcją hashującą, przy małej ilości danych zapierdala niesamowicie dobrze, generując małą ilość bajtów do której można się odwołać przy obliczeniach związanych z tymi danymi, bez korzystania z nich samych. Więc tutaj też będzie, **I chuj.**

Aby sprawdzić czy w obu folderach “synchronizujących się” są te same pliki (lub czy coś się zmieniło) wykorzystamy prosty generator hashu z masy wszystkich plików I ich listy, bo są to wartości które zawsze zmienią się jeśli coś pozamieniamy w plikach. (Chyba że zamienimy tylko kilka bajtów w danych plikach, wtedy nic nie zauważymy, ale to nie powinno mieć miejsca, bardzo słaby przypadek).

Hash_md5 = Masa folderu “sync” + Lista plików (tak jak z komendy “du”)

Dla usprawnienia możemy podczas bootu lub w randomowych momentach (albo np. Co 10 minut) wykonywać naszą funkcję pochodną od “du” na folderze sync I zrzucać jego zawartość do jakiegoś pliku w /etc/ (nie wiem jak na androidzie, wymyślcie coś) aby działać na tym pliku w odstępach czasowych ~1 minuta.

Ten hash będzie wysyłany z komendą “SK” (“Synchronizuj kurwo”) aby sprawdzić czy wszystko się zgadza. Jeśli hashe się zgadzają, drugie urządzenie odsyła komende “Yes”, jeśli nie, “No”.

Jeśli hashe z obu urządzeń się nie zgadzają, dochodzi do synchronizacji na początku samej listy plików I ich masy. Wtedy wyszukiwane są wartości które się nie zgadzają I **pliki o nowszej dacie** są wysyłane do drugiego urządzenia.