

Návrh bezpečnej videokonferenčnej architektúry

PB173 - Tematicky zaměřený vývoj aplikací v jazyce C/C++

Základná myšlienka a cieľ:

Cieľ je navrhnúť architektúru, na bezpečnú komunikáciu (s ohľadom na tajnosť - obsah a autenticitu - pôvod správy), ktorá umožní klientovi overiť svoj verejný kľúč u dôveryhodnej certifikačnej autority. Prihlásiť sa na dôveryhodný server, získať zoznam prihlásených užívateľov s možnosťou požiadať druhú stranu o zahájenie komunikácie. Udržať návrh čo najjednoduchší a zároveň otvorený pre ďalšie prípadne úpravy.

Spôsob realizácie:

Aplikácia bude rozdelená na tri samostatne bežiacie časti – Certifikačnú autoritu, Klienta, Server.

Certifikačná autorita – má na starosti overenie verejného kľúča klienta. Klient po vygenerovaní dvojice – súkromní/verejný kľúč (RSA). Pošle certifikačnej autorite klient svoj verejný kľúč na podpis, pre zjednodušenie neprebíha, žiadne overovanie totožnosti klienta. Takto podpísaný kľúč neskôr považuje server za dôveryhodný. Klient sa môže pomocou API servera zaregistrovať do zoznamu aktívnych užívateľov, pri registrácii sa overí, či je verejný kľúč klienta podpísaný súkromným kľúčom certifikačnej autority, klient si zvolí prihlasovacie meno a heslo. Heslo sa na strane klienta zahashuje spolu s náhodnou hodnotou, náhodná hodnota sa uloží na strane klienta, pre ďalšie použitie a výsledná hash sa pošle serveru, kde sa uloží spolu s menom a verejným kľúčom. Takto server pozná iba hash hesla s jedinečnou náhodnou hodnotou pre daný server. Po úspešnom zaregistrovaní sa klient môže prihlásiť na server a vyžiadať si zoznam aktívnych užívateľov. Klient môže cez server požiadať ďalšieho klienta o komunikáciu. Server mu odošle požiadavku s prípadnou adresou klienta v sieti s ktorým chce komunikovať a jeho verejným kľúčom. Ak má klient o komunikáciu záujem, stanoví spoločný kľúč k symetrickej šifre (AES_CGM), ktorá zabezpečí, že správa bude dôverná a nepozmenená. Tento kľúč zašifruje verejným kľúčom klienta a pošle mu ho, spolu so svojou adresou. Takto môžu ustaviť spoločnú šifrovanú komunikáciu. O tom či klient prijať požiadavku, alebo či komunikácia pokračuje, nemá server žiadnu informáciu. Tým predchádzame aj vyťažovaniu servera, ak by komunikáciu obstarával on. O ukončení komunikácie dá jeden klient druhému vedieť pomocou hlavičky v správe. Nerátame s osobitnou funkciou v API.

Možné zraniteľnosti, ktoré neriešime (zatiaľ):

- dôveryhodnosť servera a certifikačnej autority (veríme im)
- zoznam klientov je verejný, stačí byť overení CA, zaregistrovať a prihlásiť sa na server
- pri prihlásení klienta k serveru neoverujeme či nie je jeho verejný kľúč vyhlásený za neplatný od CA

Jednotlivé API:

všetky funkcie majú predponu HK_ (Hrochokobry), ktorá je zároveň duševným vlastníctvom autorov :)

Certifikačná autorita (CA) -

Tasks of CA:

1. Signing public keys of clients(at the moment w/o checking for client identity)
2. (Might be added later) Revoking compromised public keys

HK_registration(...)

/**

* function to verify new client and sign his public key

```
* @char * par name
* @char * public_key
**/
```

Client -

Tasks of client:

1. Accept request for communication(may accept – communication starts, deny – nothing happens)
2. Receive data – read and process received data

HK_request_for_communication(...)

```
/**
* receive request for communication from another client, has two possibilities reject or
* accept, then must set up symmetric key, sign it by public key of another client and send
* @client* client_name
* @char* address
**/
```

HK_receive_data(...)

```
/**
* function to receive data
**/
```

Server-

Tasks of server:

1. Register and check identity of new users(with help from CA)
2. After successful registration add user to list(name, password, public key)
3. When user signs in server sends him list of other users
4. Establish connection between users (exchange of IP addresses, rest of communication is not lead through server)
5. Checking for online users = actualisation of list of users so it is not beign sent whole

HK_registration(...)

```
/**
* @brief verifies data and afterwards adds the user to list
* @param username
* @param password - +salted and hashed
* @param public key
*/
```

HK_login(...)

```
/**
* @brief logs user in
* @param username
* @param password - +salted and hashed
*/
```

HK_get_user_list(...)

```
/**
* @brief sends list of users, possible to access only if requester is logged in
*/
```

HK_request_for_communication(...)

```
/**
```

* @brief one client requests for communication with another, requester must be logged in
* @param target name of the client which should be asked for communication
*/