**HTTP/1.1 - Authentication**

HTTP/1.1 е версия на протокола за хипертекстов трансфер (HTTP), който служи за комуникация между уеб браузъри и уеб сървъри. Важни елементи от тази версия включват начините за удостоверяване и авторизация, както и сигурността на преноса на данни. Нека разгледаме някои от ключовите аспекти, описани в посочените RFC документи:

* RFC 7235 - HTTP/1.1 Authentication
* RFC 7235 обновява и разширява дефинициите за удостоверяване в HTTP:

1. Автентикация отнася до процеса, при който сървърът проверява идентичността на клиента. Тя осигурява, че клиентът е този, за когото се представя.
2. Авторизация се отнася до процеса на определяне на правата на достъп до ресурси за потвърден потребител.

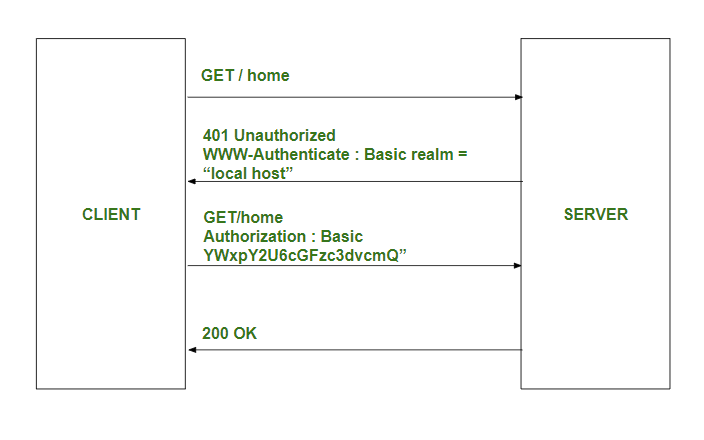
**Основни методи за автентикация:**

* **Basic Authentication**

Basic Authentication е прост метод за удостоверяване в HTTP, който използва стандартни заглавки за автентикация, за да предава потребителски имена и пароли. При този метод, потребителското име и паролата се комбинират в един стринг "username:password", който след това се кодира с Base64. Полученият кодиран стринг се изпраща в заглавката Authorization на HTTP заявката, предварен от думата Basic.

Пример:

Ако потребителското име е "user" и паролата е "pass", комбинираната стойност "user:pass" се кодира в Base64 до нещо като "dXNlcjpwYXNz".



**Сигурност**:

Тъй като креденциалите се изпращат просто кодирани, а не криптирани, Basic Authentication е уязвима към мрежови подслушвания. Човек, който прехваща трафика, лесно може да декодира Base64 и да види потребителското име и паролата.

За повишаване на сигурността, Basic Authentication трябва да се използва в комбинация с HTTPS, което осигурява криптиране на данните при трансфера им.

* **Digest Authentication**

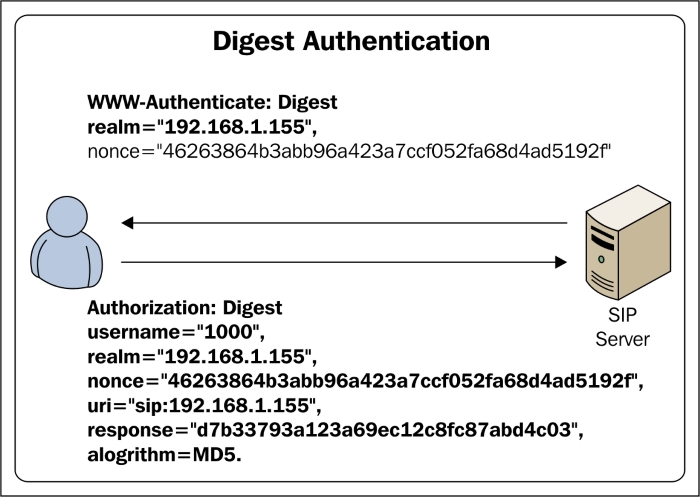
Digestive Authentication е по-сигурен метод за удостоверяване, който използва хеширане, за да избегне изпращането на пароли в явен текст. Вместо да изпраща реалните потребителско име и парола, клиентът изпраща хеш версия на паролата заедно с други допълнителни данни.

Процес на удостоверяване:

1.Сървърът първо изпраща "предизвикателство" до клиента, което включва нонс (еднократен номер, използван за текущата сесия) и други параметри.

2.Клиентът използва тези параметри, заедно с потребителското име и парола, за да генерира хеш. Този хеш, наречен отговор, се изпраща обратно на сървъра.

3.Сървърът проверява хеша, използвайки същите параметри, и ако хешовете съвпадат, клиентът се счита за удостоверен.



**Сигурност**:

Digest Authentication предлага подобрена сигурност спрямо Basic Authentication, тъй като паролата никога не се изпраща по мрежата в явен текст.

Въпреки това, методът все още има своите слабости, например, податливостта на атаки чрез възпроизвеждане, ако нонсът не се контролира правилно.

Digest метода имат своите приложения, но важно е да се вземат предвид потенциалните рискове и да се прилагат подходящи мерки за сигурност, особено в чувствителни приложения.

* **RFC 6750 - The OAuth 2.0 Authorization Framework: Bearer Token Usage**

Bearer Tokens:

* Тип токени, използвани за достъп до ресурси без да се налага да изпращате повторно креденциалите.
* Трябва да се използват само по защитени канали, тъй като притежателят на токена може да бъде удостоверен само чрез предоставянето на токена.

* **Proxy Authentication**

Процесът, при който потребителите трябва да се удостоверят при прокси сървъра, преди техните заявки за интернет ресурси да бъдат обработени. Това е специфичен тип автентикация, който се изисква, когато прокси сървъри се използват за регулиране на достъпа до мрежата или за увеличаване на сигурността чрез филтриране на трафика.

Как работи Proxy Authentication?

Когато клиент изпрати заявка към интернет, прокси сървърът може да блокира заявката и да поиска от клиента да се удостовери. Това се прави чрез изпращане на статусен код 407 (Proxy Authentication Required) обратно към клиента, съпроводен със заглавка Proxy-Authenticate, която описва как клиентът може да се удостовери.

Пример за Proxy Authentication:

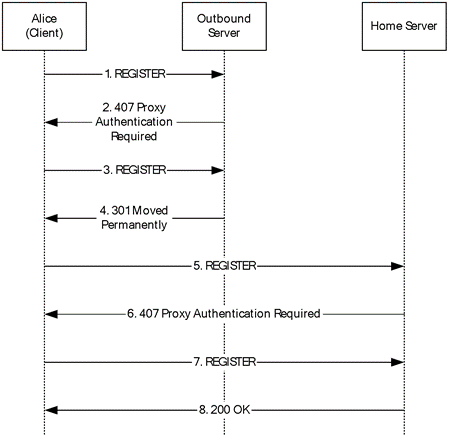
1.Клиентът изпраща HTTP заявка към прокси сървър.

2.Прокси сървърът отговаря с HTTP 407 Proxy Authentication Required.

3.Заглавката Proxy-Authenticate може да включва типа на удостоверяване (например Basic или Digest) и областта на удостоверяване (realm), която дава указание къде се прилага удостоверяването.

4.Клиентът трябва да изпрати своите удостоверителни данни, като обикновено ги кодира (Base64 за Basic, специфичен хеш за Digest).

5.Прокси сървърът проверява получените креденции и, ако са валидни, позволява достъп до мрежовите ресурси.



Настройка и сигурност:

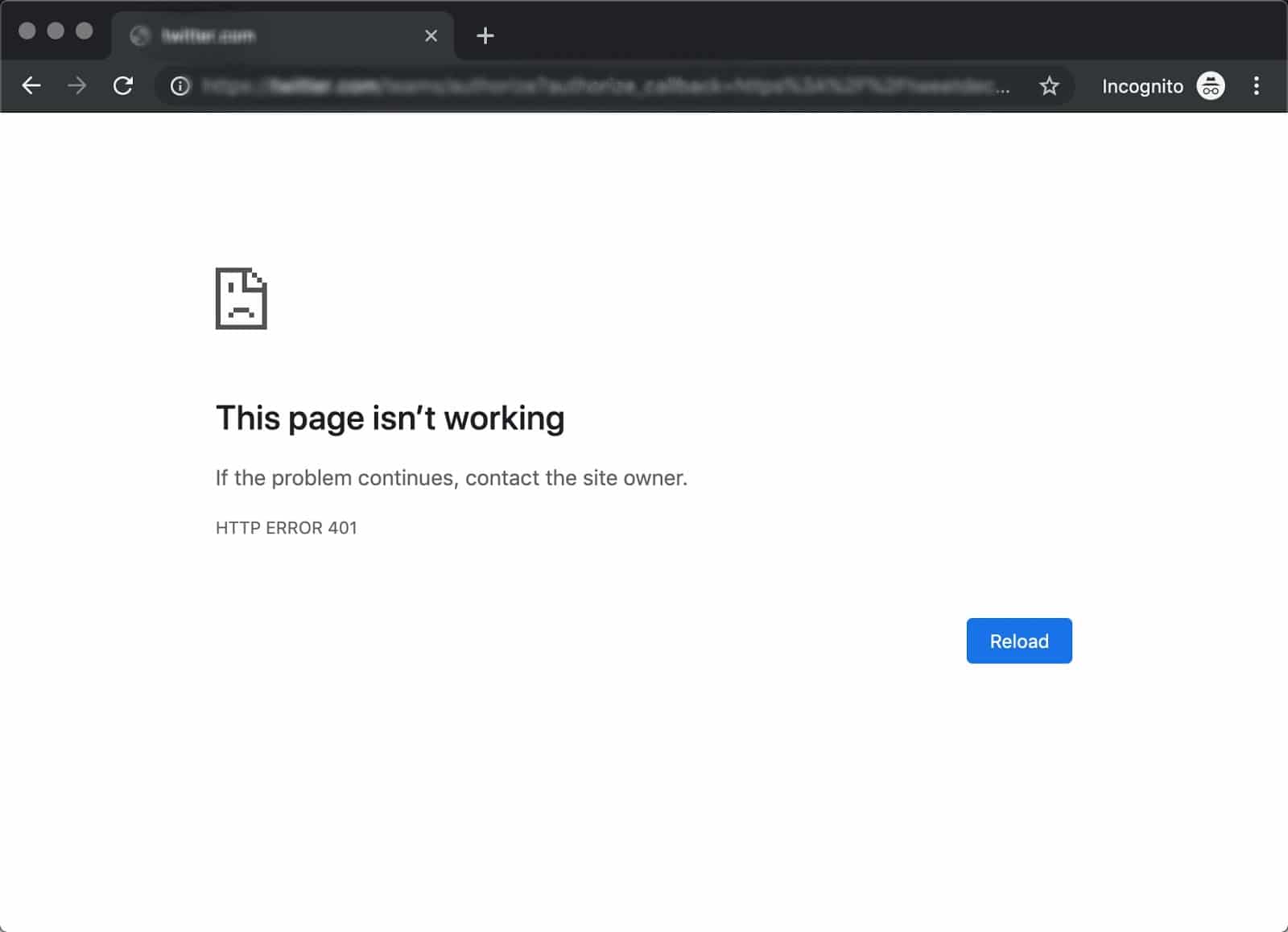
* Конфигурация: Администраторите на мрежата настройват прокси сървъра да изисква удостоверяване, което помага за контролиране на достъпа до мрежови ресурси и управление на трафика.
* Сигурност: Proxy Authentication добавя допълнителен слой сигурност, като осигурява, че само удостоверени потребители могат да достъпват мрежови ресурси. Това може да помогне за предотвратяване на неоторизиран достъп и атаки.
* Данни за удостоверяване: Както при всички видове удостоверяване, много е важно защитените канали като HTTPS да се използват при предаването на удостоверителни данни, за да се предотврати тяхното залавяне от неоторизирани лица.

Proxy Authentication е особено полезна в корпоративни и образователни среди, където контролът на мрежовия достъп и сигурността са от съществено значение.

Основни HTTP Грешки Свързани с Автентикация:

HTTP статусните кодове са част от отговорите, които уеб сървърите изпращат към клиентите (например уеб браузъри), за да укажат как е обработена тяхната заявка. Ето някои от най-често срещаните HTTP грешки свързани с автентикация и авторизация:

1. 401 Unauthorized



HTTP 401 статусният код се изпраща, когато дадена заявка изисква удостоверяване, но потребителят не се е удостоверил или предоставените креденциали не са валидни за достъп до ресурса. Сървърът може да включи заглавка WWW-Authenticate с информация за това как клиентът може да се удостовери правилно.

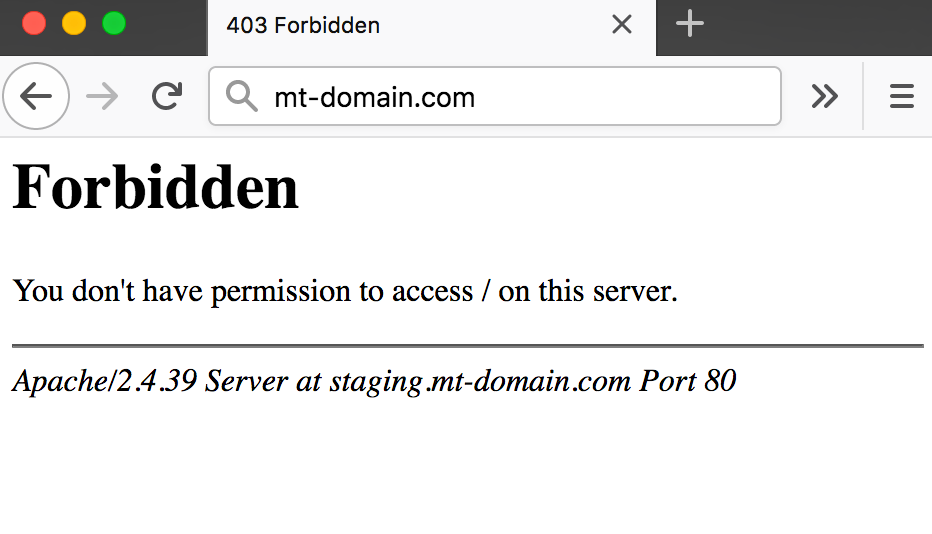
Пример:

Когато потребителят достъпва защитена част от уебсайт без да е предоставил правилни креденциали, сървърът ще върне:

HTTP/1.1 401 Unauthorized

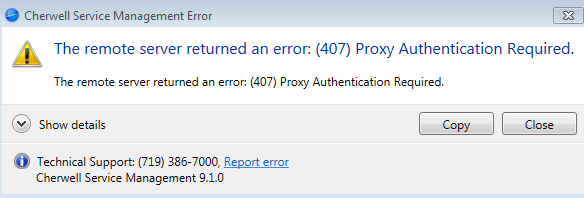
WWW-Authenticate: Basic realm="Private Area"

2. 403 Forbidden



Кодът 403 Forbidden се връща, когато сървърът разбира заявката, но отказва да я обслужи поради правила, които забраняват на този потребител достъпа до ресурса. Това не е проблем на удостоверяването, а на авторизацията. Дори и потребителят да е удостоверен, той може да получи 403, ако няма нужните права за достъп.

3. 407 Proxy Authentication Required



Кодът 407 се използва, когато необходимостта от удостоверяване идва от прокси сървър, който между клиента и интернет. Този код е подобен на 401, но указва, че клиентът трябва да се удостовери с прокси сървъра, преди заявката да продължи.

Сигурност:

При използване на HTTP, особено с механизми за автентикация, винаги се препоръчва използването на HTTPS за криптиране на трафика и защита на креденциалите от мрежови подслушвания.

Безопасната конфигурация и управление на сесиите са от съществено значение за предотвратяване на атаки като man-in-the-middle и replay атаки.

### HTTPS (Hypertext Transfer Protocol Secure)

HTTPS е разширение на HTTP (Hypertext Transfer Protocol), което е основен протокол, използван за обмен на информация в интернет. HTTPS добавя слой сигурност чрез криптиране на данните, които се изпращат между уеб браузера и уеб сървъра. Това предотвратява атаки като "man-in-the-middle", където някой може да подслушва или модифицира данните по време на тяхната трансмисия.

### SSL и TLS

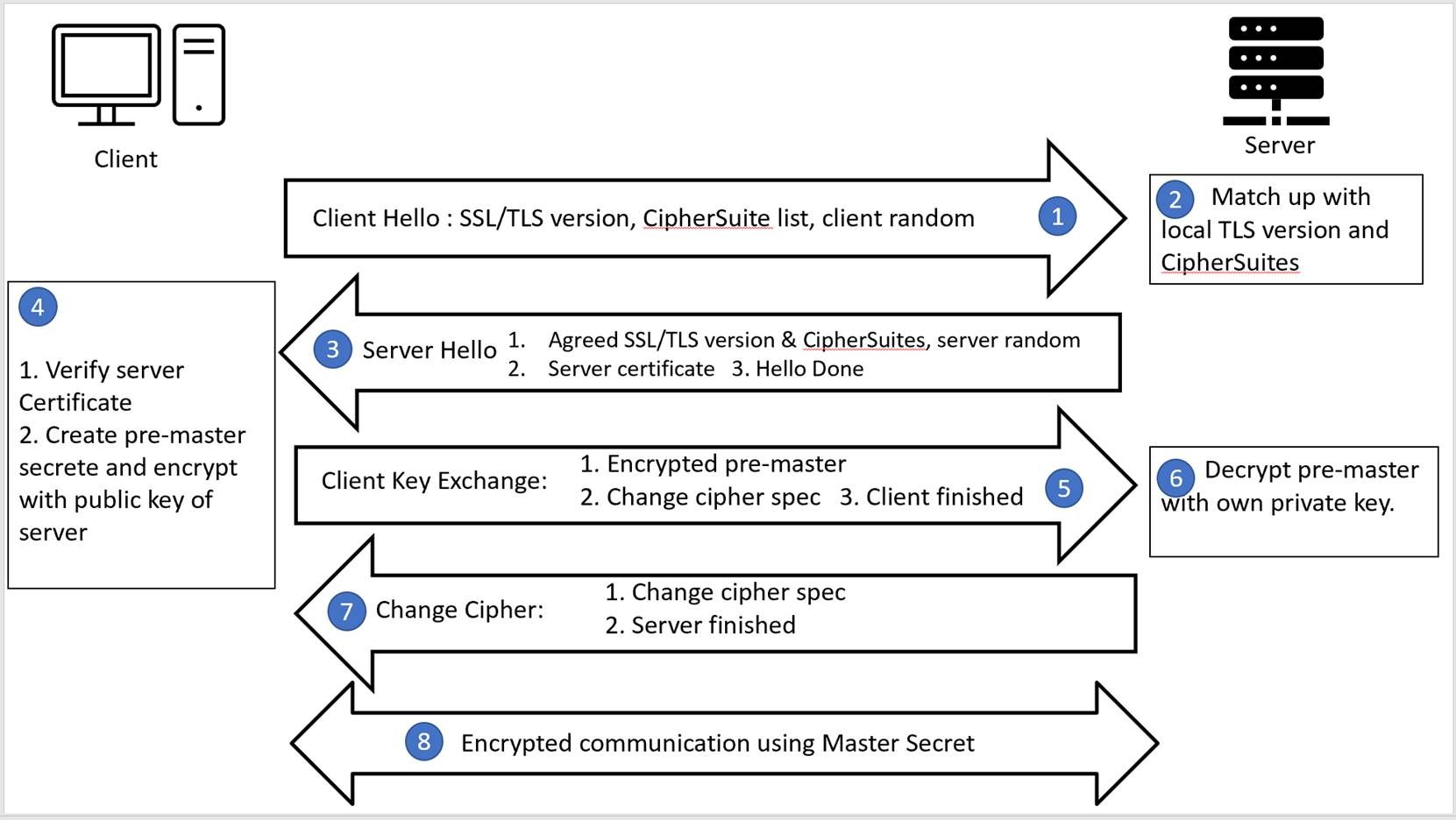
Ето основните аспекти на SSL и TLS:

1. **Криптиране** - Поддържа конфиденциалността на данните, като ги криптира преди изпращането и декриптира след получаването. Това гарантира, че данните не могат да бъдат прочетени или модифицирани по време на трансфера.
2. **Аутентикация** - Поддържа аутентикацията на сървъра (и понякога и на клиента) чрез използване на сертификати. Сертификатите са издадени от сертификационни органи (CA), които удостоверяват идентичността на сървъра.
3. **Целостност** - Проверява целостта на данните чрез хеш функции и MAC (Message Authentication Code) за да се увери, че данните не са били променени по време на трансмисия.

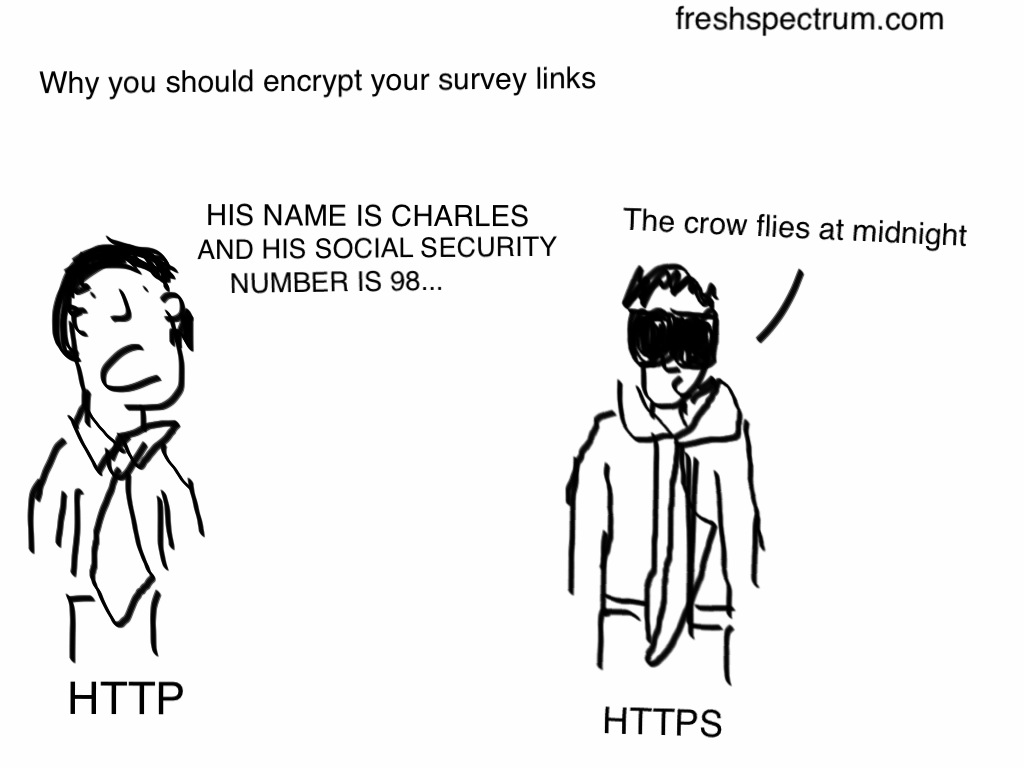
Процес на установяване на връзка

Процесът на установяване на защитена връзка чрез TLS/SSL се нарича TLS/SSL "handshake" и включва следните стъпки:

1. **Начална връзка** - Клиентът и сървърът се съгласяват на версията на протокола, криптографичните алгоритми и други настройки, необходими за сигурна връзка.
2. **Аутентикация и ключов обмен** - Сървърът изпраща своя сертификат на клиента. Клиентът проверява сертификата с помощта на издаващите сертификационни органи. След успешна проверка, клиентът и сървърът генерират сесийни ключове за криптиране на данните.
3. **Завършване на свързването** - Връзката е сигурна, и двамата могат да започнат да изпращат криптирани данни.



Просто ползвай HTTPS :



Използвани източници :

[1] “Hypertext Transfer Protocol (HTTP/1.1): Authentication”:

<https://tools.ietf.org/html/rfc7235> Последно посетен : 16.04.2024

[2]”The OAuth 2.0 Authorization Framework: Bearer Token Usage”

<https://tools.ietf.org/html/rfc6750> Последно посетен : 13.04.2024

[3]Blog by : Vivian Seixas published on 20.06.2023” What Is HTTP Authentication?

”https://www.azion.com/en/blog/what-is-http-authentication/ Последно посетен : 16.04.2024

[4] HTTP Authentication

<https://http.dev/authentication> Последно посетен : 14.04.2024

[5] “HTTP Authentication: Basic and Digest Access Authentication”

<https://datatracker.ietf.org/doc/html/rfc2617> Последно посетен : 16.04.2024

[6]Избираема дисциплина водена от доц.Георги Георгиев "Основи на ТСР/IP (в.4 и в.6)" : "Module 11: Implementing Certificate Services"

https://learn.fmi.uni-sofia.bg/pluginfile.php/467750/mod\_resource/content/1/TCP\_IP Fundamentals - Module 11.htm Последно посетен : 15.04.2024