

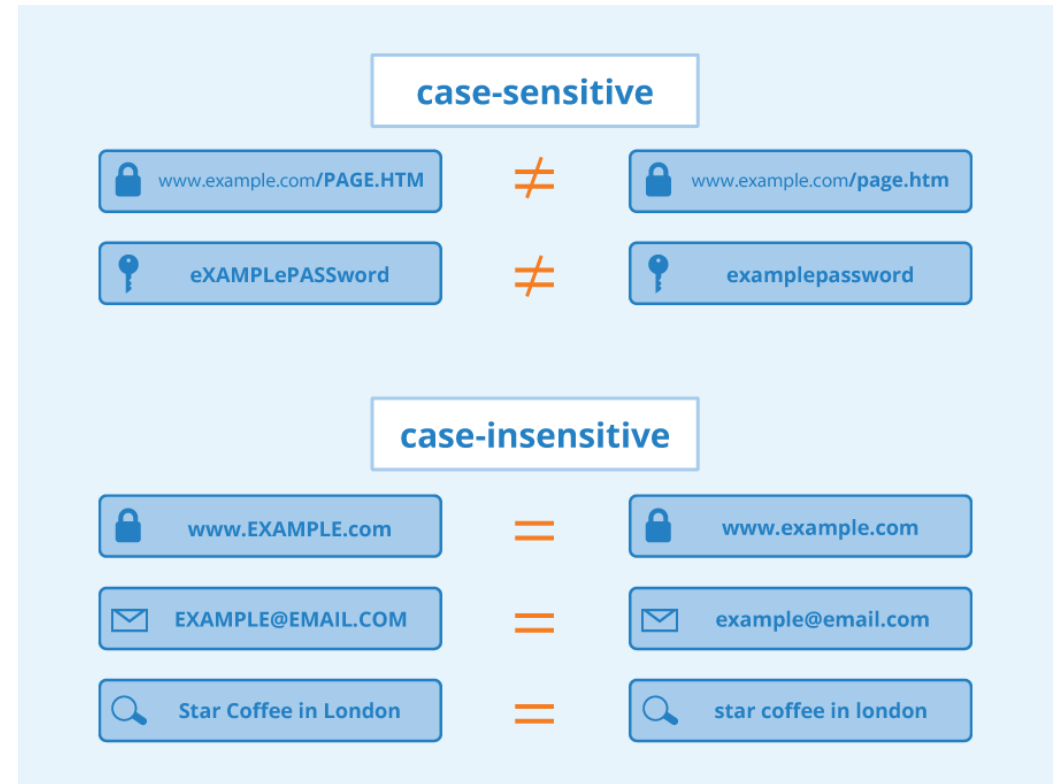
Yazılım güvenliği

İPUÇLARI

Kimlik doğrulama

1. Case-(in)sensitive kullanıcı adı

- Benzersiz kullanıcı adları
- Hızlı erişim
- Hata mesajları



Kimlik doğrulama

2. Yetkili/iç hesapların front-end'de kullanılmaması

- Root, DBA, support
- DMZ ?!?

3. Parola uzunluğu , karmaşıklığı, güncellemesi

Admins cannot login from here.

My account

Login

Username or email address *

admin

Password *

.....

☐ Remember me

Log in

[Lost your password?](#)

Register

Email address *

admin

Password *

.....

Very weak - Please enter a stronger password. 😞

Hint: The password should be at least twelve characters long. To make it stronger, use upper and lower case letters, numbers, and symbols like ! " ? \$ % ^ & .

Your personal data will be used to support your experience throughout this website. To manage or delete your data, you may visit your [privacy settings](#).

Kimlik doğrulama

4. Parola güncelleme
güvenli bir kanaldan

5. Parolalar DB'de

hash

salt

pepper

The screenshot shows the e-Devlet Kapısı Kimlik Doğrulama Sistemi (e-Government Gate Identity Verification System) interface. The browser address bar shows the URL giris.turkiye.gov.tr/Giris/SifreSifirlama. The page title is "e-Devlet Kapısı Kimlik Doğrulama Sistemi". The interface includes a navigation bar with "Şifremi Unuttum" and "İletişim Bilgisi" links. Below the navigation bar is a breadcrumb trail: "Başlangıç > Kimlik No. > Kimlik Detay > İletişim Bilgisi > Güvenlik Kodu > Şifre Oluşturma > Bitiş". A message states: "Nüfus bilgileriniz doğrulanmıştır... Lütfen profilinizde kayıtlı iletişim bilgilerinizi giriniz. Girdiğiniz bilgilerin profilinizde kayıtlı olanlar ile bire bir uyuşması gerekmektedir. Aksi takdirde işlem iptal edilecektir." The form contains two main sections: "Cep Telefonu" (Mobile Phone) and "e-Posta Adresi" (Email Address). The "Cep Telefonu" section includes a dropdown for "Ülke Kodu" (Country Code) set to "TÜRKİYE (90)" and a text input for "Telefon Numarası" (Phone Number). Below this is a note: "Örneğin Ülke Kodu=90, Telefon=5XXXXXXX". The "e-Posta Adresi" section has a text input for the email address. At the bottom of the form are two buttons: "İptal Et ve Geri Dön" (Cancel and Go Back) and "Devam Et" (Continue). The footer includes the copyright notice "© 2022, Ankara - Tüm Hakları Saklıdır" and the text "Gizlilik ve Güvenlik Hızlı Çözüm Merkezi".

Kimlik

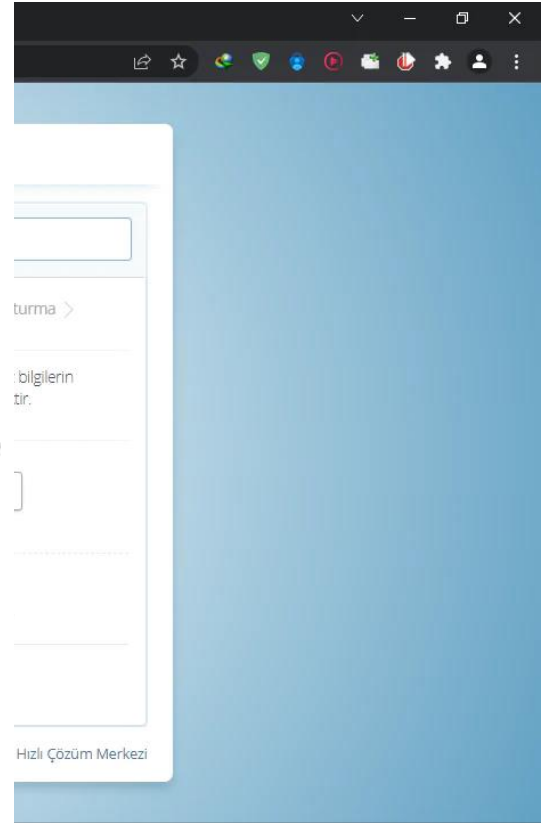
4. Parola güveni

URL Tokens

URL tokens are passed in the query string of the URL, and are typically sent to the user via email. The basic overview of the process is as follows:

1. Generate a token to the user and attach it in the URL query string.
2. Send this token to the user via email.
3. Don't rely on the `Host` header while creating the reset URLs to avoid `Host Header Injection` attacks. The URL should be either be hard-coded, or should be validated against a list of trusted domains.
4. Ensure that the URL is using HTTPS.
5. The user receives the email, and browses to the URL with the attached token.
6. Ensure that the reset password page adds the `Referrer Policy` tag with the `noreferrer` value in order to avoid `referrer leakage`.
7. Implement appropriate protection to prevent users from brute-forcing tokens in the URL, such as rate limiting.
8. If required, perform any additional validation steps such as requiring the user to answer `security questions`.
9. Let the user create a new password and confirm it. Ensure that the same password policy used elsewhere in the application is applied.

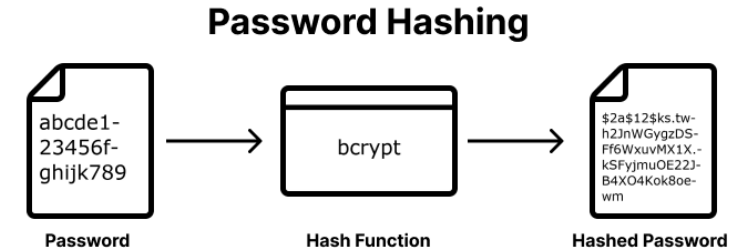
Note: URL tokens can follow on the same behavior of the `PINs` by creating a restricted session from the token. Decision should be made based on the needs and the expertise of the developer.



Kimlik doğrulama

6. Doğrulama için zaman

- sabit süreli algoritmalar



```
def check_password(password):  
    correct = "hunter2"  
    for i in range(len(password)):  
        if i >= len(correct) or password[i] != correct[i]:  
            return False  
    return len(password) == len(correct)
```

Figure 1. Example code vulnerable to a timing attack

Kimlik doğrulama

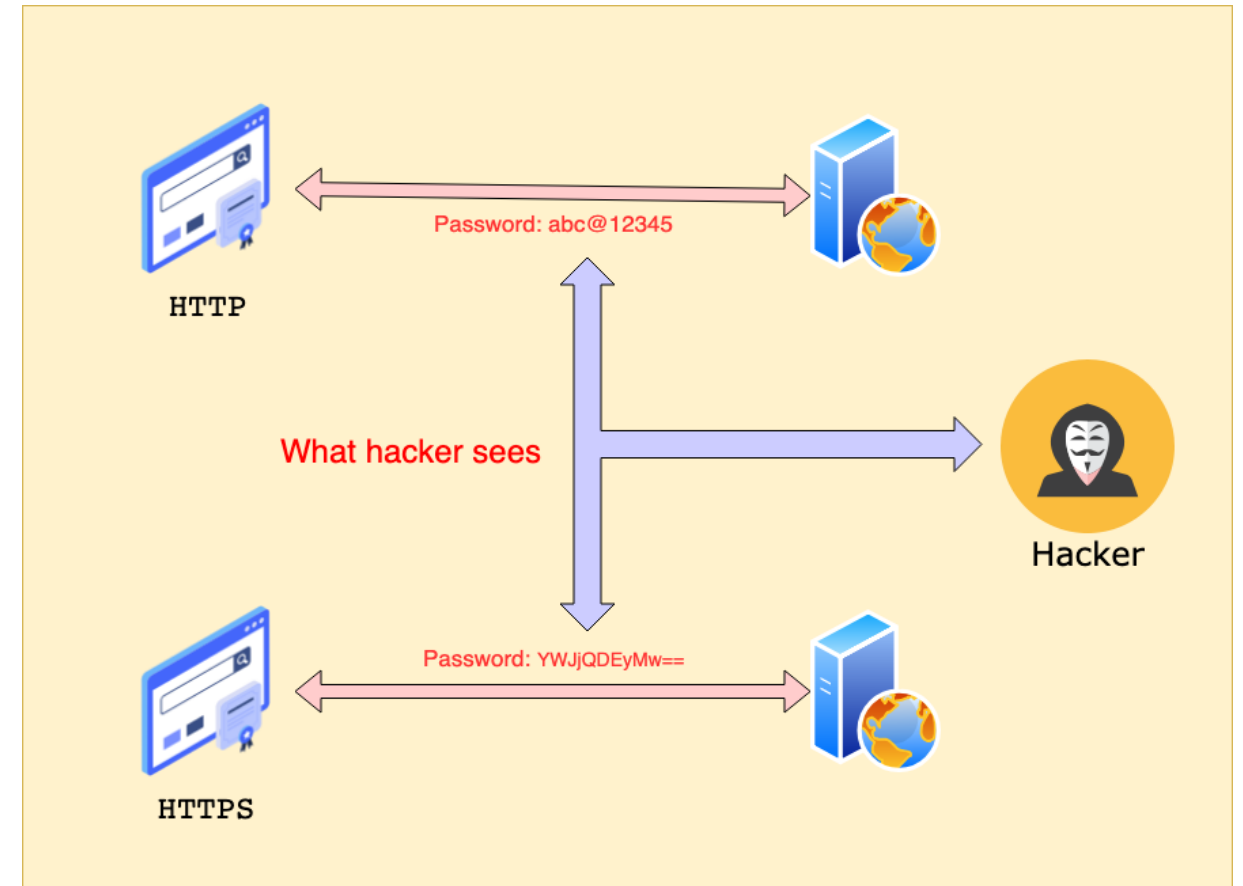
7. TLS giriş bilgisi

8. Gerekli /hassas durumlarda
tekrar kimlik doğrulama

Reauthentication

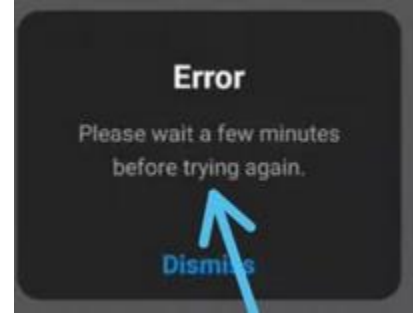


Reauthentication is a cybersecurity concept that refers to the process of requiring users to offer additional verification or authentication credentials to regain their access to a system, application, or account that they have previously logged into.



Kimlik doğrulama

9. Genel hata mesajları



500. That's an error.

The server encountered an error and could not complete your request.

If the problem persists, please [report](#) your problem and mention this error message and the query that caused it. That's all we know.



```
Warning: file_put_contents(/home/admin/web/mywebserver.example.com/public_html/site/assets/cache/FileCompiler/site/templates/_init.php): failed to open stream: Permission denied in /home/admin/web/mywebserver.example.com/public_html/wire/core/FileCompiler.php on line 389

Warning: fopen(/home/admin/web/mywebserver.example.com/public_html/site/assets/logs/file-compiler.txt): failed to open stream: Permission denied in /home/admin/web/mywebserver.example.com/public_html/wire/core/FileLog.php on line 82

Warning: file_put_contents(/home/admin/web/mywebserver.example.com/public_html/site/assets/cache/FileCompiler/site/templates/_init.php): failed to open stream: Permission denied in /home/admin/web/mywebserver.example.com/public_html/wire/core/FileCompiler.php on line 389

Warning: fopen(/home/admin/web/mywebserver.example.com/public_html/site/assets/logs/file-compiler.txt): failed to open stream: Permission denied in /home/admin/web/mywebserver.example.com/public_html/wire/core/FileLog.php on line 82

Warning: file_put_contents(/home/admin/web/mywebserver.example.com/public_html/site/assets/cache/FileCompiler/site/templates/_init.php): failed to open stream: Permission denied in /home/admin/web/mywebserver.example.com/public_html/wire/core/FileCompiler.php on line 389

Warning: fopen(/home/admin/web/mywebserver.example.com/public_html/site/assets/logs/file-compiler.txt): failed to open stream: Permission denied in /home/admin/web/mywebserver.example.com/public_html/wire/core/FileLog.php on line 82

Warning: file_put_contents(/home/admin/web/mywebserver.example.com/public_html/site/assets/cache/FileCompiler/site/templates/_init.php): failed to open stream: Permission denied in /home/admin/web/mywebserver.example.com/public_html/wire/core/FileCompiler.php on line 389
```



SİBER GÜVENLİK EĞİTİM PORTALI

[Ana Sayfa](#)

[Portal Hakkında](#)

[Eğitimler ▾](#)

[Uzmanlık Alanları ▾](#)

[Rehberler ▾](#)

[Blog Yayınları](#)

Hata - Call to undefined method cachestore_dummy::find_by_prefix()

[Bu hata hakkında daha fazla bilgi](#)

API Güvenliği

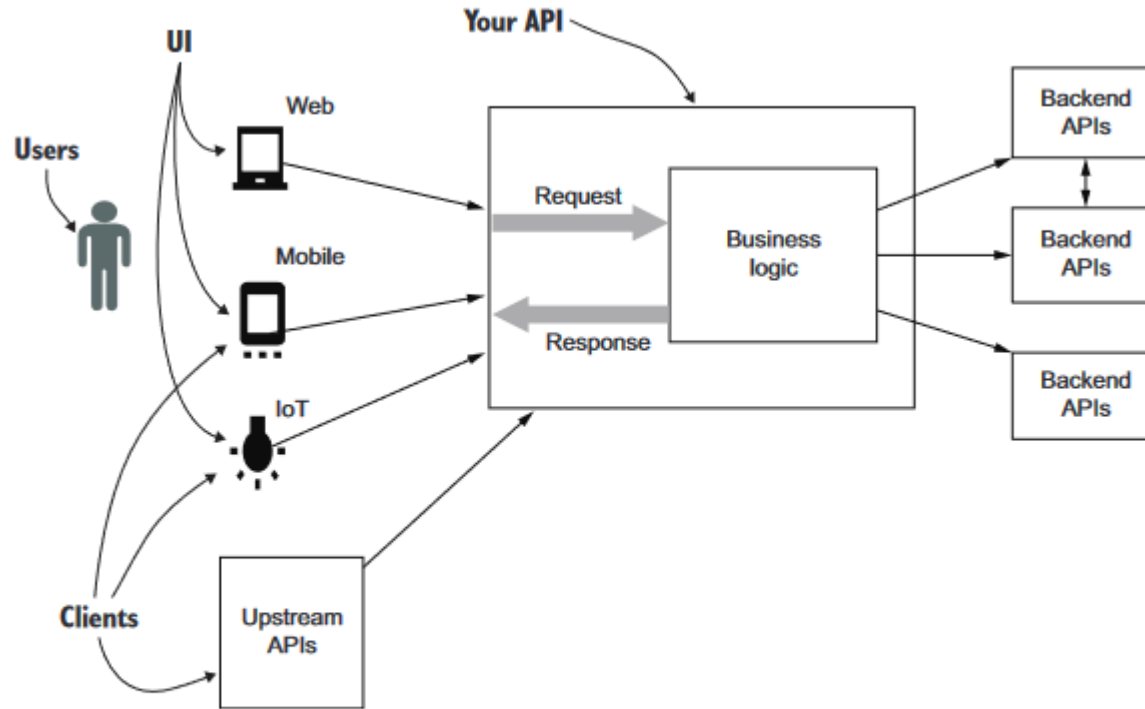


Figure 1.1 An API handles requests from clients on behalf of users. Clients may be web browsers, mobile apps, devices in the Internet of Things, or other APIs. The API services requests according to its internal logic and then at some point returns a response to the client. The implementation of the API may require talking to other “backend” APIs, provided by databases or processing systems.

API Güvenliği

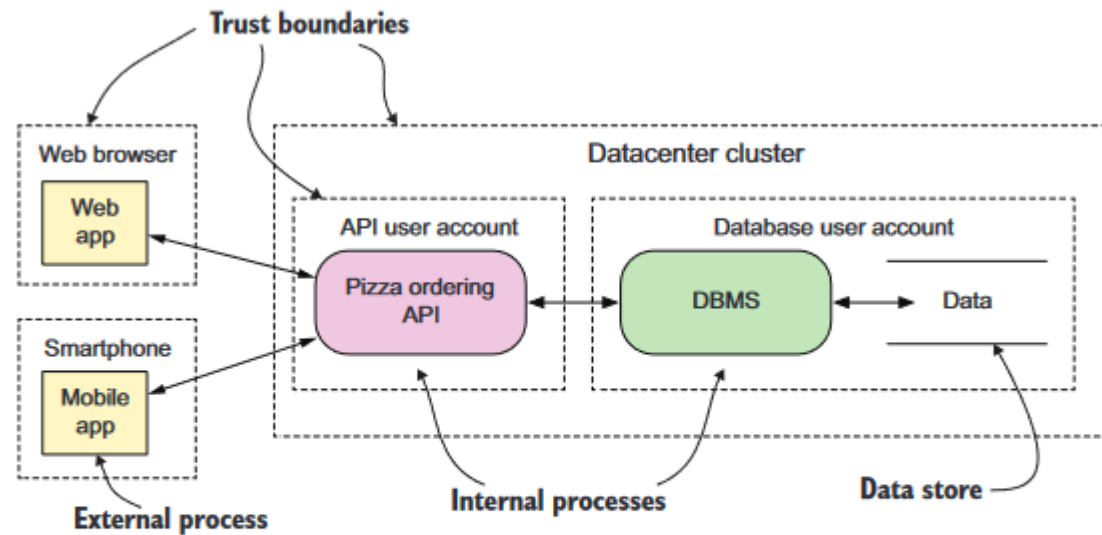


Figure 1.6 An example dataflow diagram, showing processes, data stores and the flow of data between them. Trust boundaries are marked with dashed lines. Internal processes are marked with rounded rectangles, while external entities use squared ends. Note that we include both the database management system (DBMS) process and its data files as separate entities.

API Güvenliği

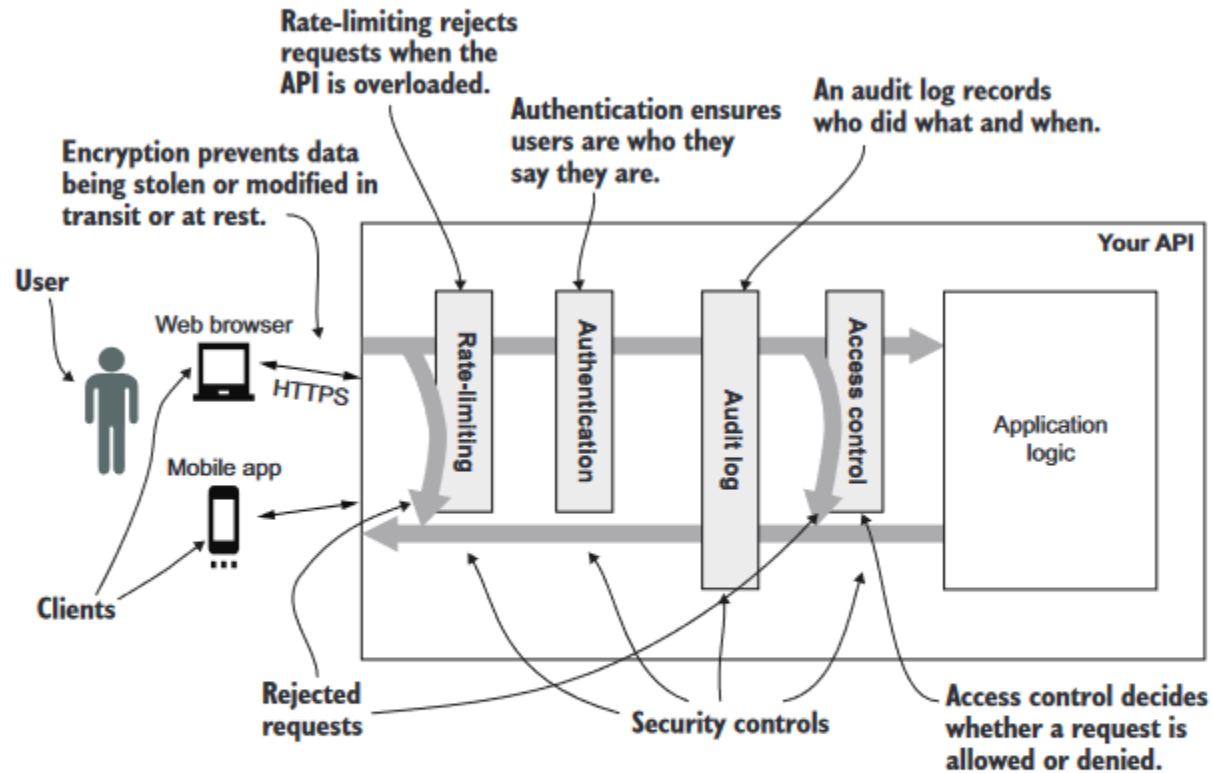


Figure 1.7 When processing a request, a secure API will apply some standard steps. Requests and responses are encrypted using the HTTPS protocol. Rate-limiting is applied to prevent DoS attacks. Then users and clients are identified and authenticated, and a record is made of the access attempt in an access or audit log. Finally, checks are made to decide if this user should be able to perform this request. The outcome of the request should also be recorded in the audit log.

API Güvenliği

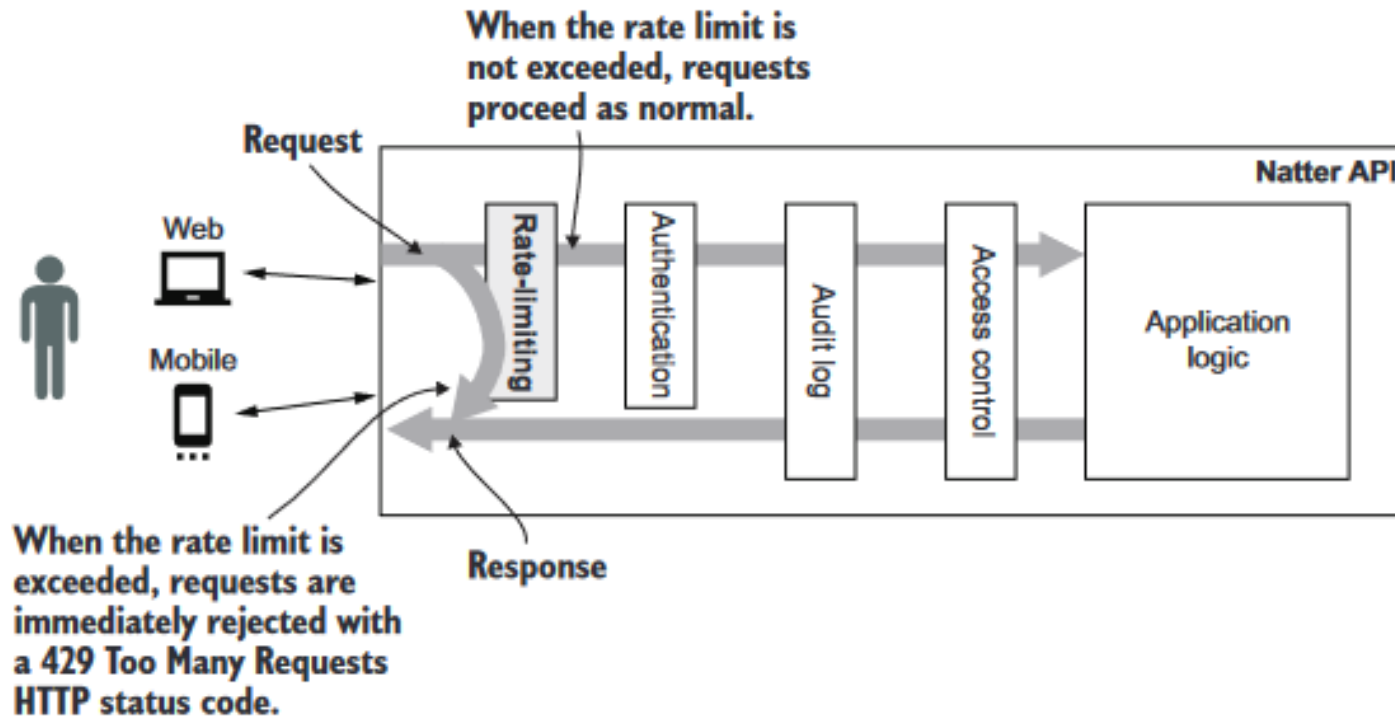


Figure 3.3 Rate-limiting rejects requests when your API is under too much load. By rejecting requests early before they have consumed too many resources, we can ensure that the requests we do process have enough resources to complete without errors. Rate-limiting should be the very first decision applied to incoming requests.

API Güvenliği

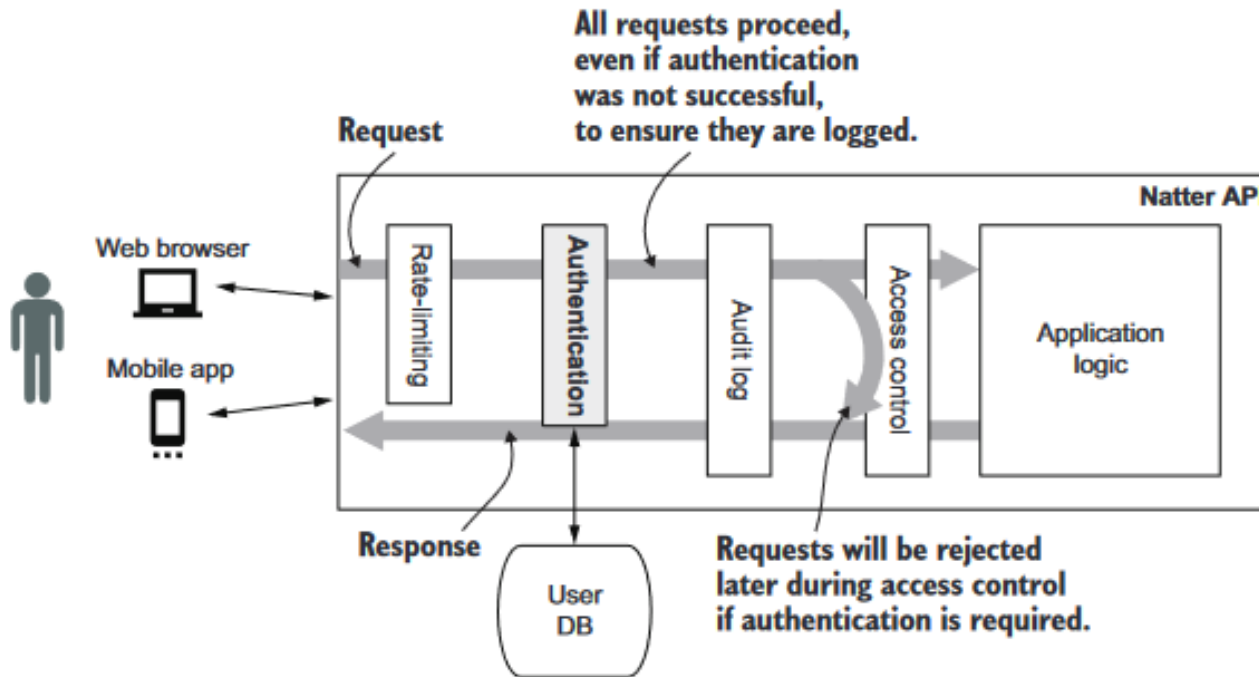


Figure 3.4 Authentication occurs after rate-limiting but before audit logging or access control. All requests proceed, even if authentication fails, to ensure that they are always logged. Unauthenticated requests will be rejected during access control, which occurs after audit logging.

API Güvenliği

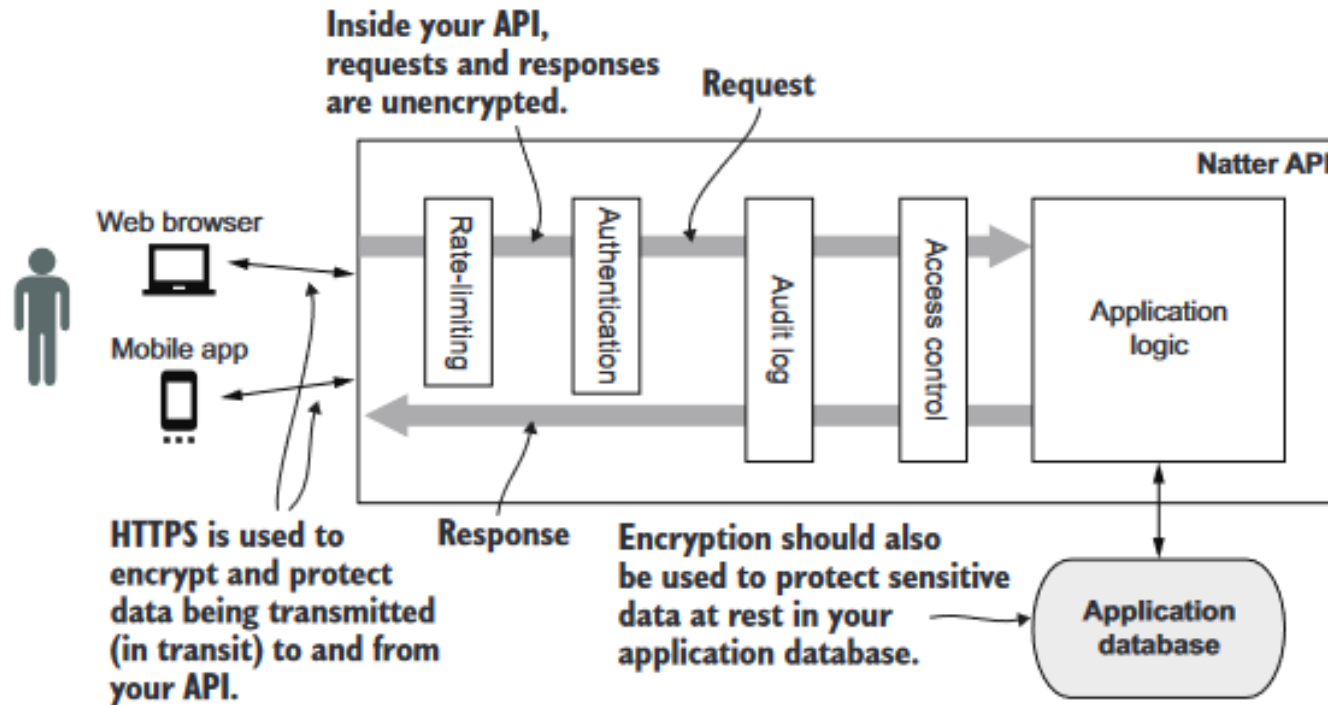


Figure 3.5 Encryption is used to protect data in transit between a client and our API, and at rest when stored in the database.

API Güvenliği

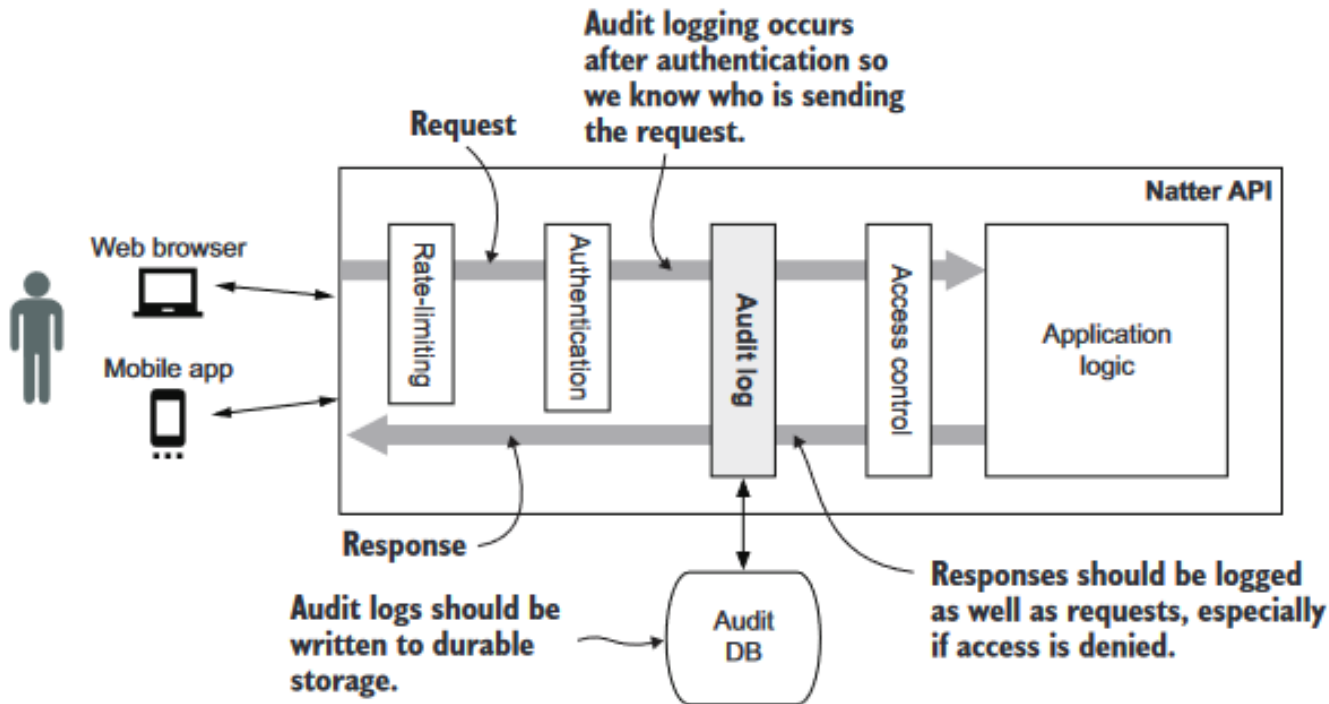


Figure 3.6 Audit logging should occur both before a request is processed and after it completes. When implemented as a filter, it should be placed after authentication, so that you know who is performing each action, but before access control checks so that you record operations that were attempted but denied.

API Güvenliği

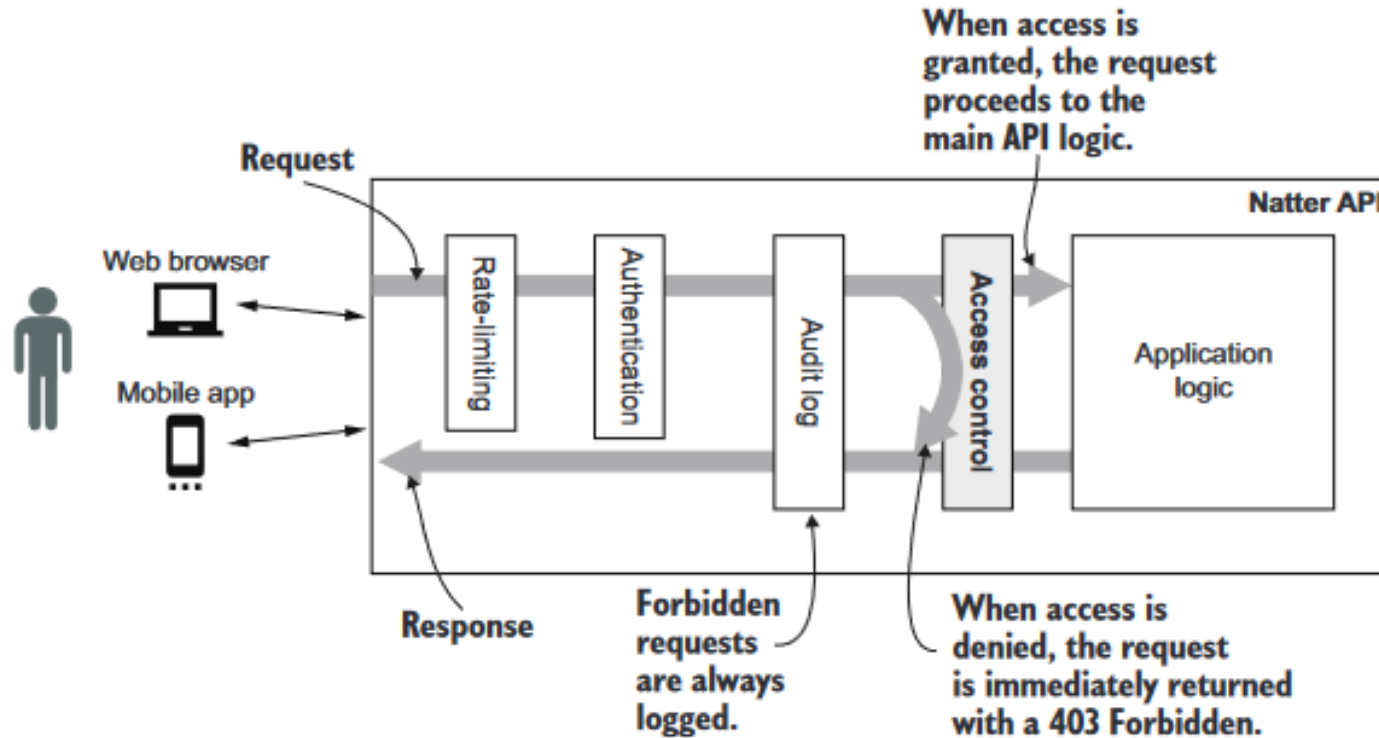


Figure 3.7 Access control occurs after authentication and the request has been logged for audit. If access is denied, then a forbidden response is immediately returned without running any of the application logic. If access is granted, then the request proceeds as normal.