



OWASP

Open Web Application
Security Project

OWASP API Top Ten

Eng. Mohammad Khreesha

Twitter : @banyrock

Facebook : @khreesha

What is OWASP?

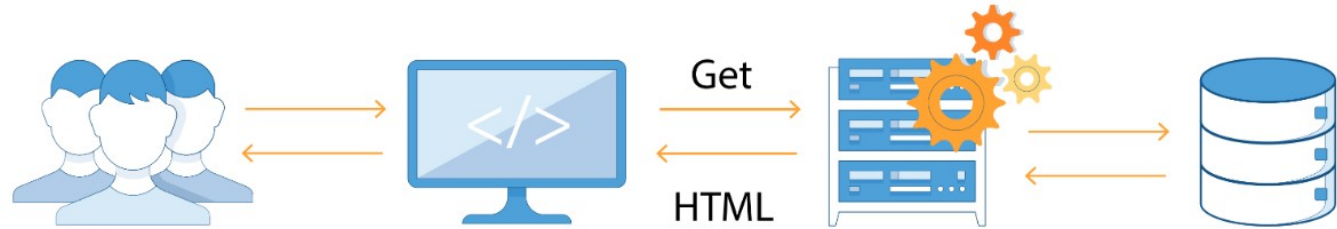
- Open Web Application Security Project (OWASP) is a nonprofit foundation that works to improve the security of software.
- A website: <http://www.owasp.org>
- A bunch of cool tools: Zed Attack Proxy, Juice Shop, Proactive Controls, Software Assurance Maturity Model (SAMM), Application Security Verification Standard (ASVS)
- A global community of like-minded people, meetups and conferences

OWASP API Top 10

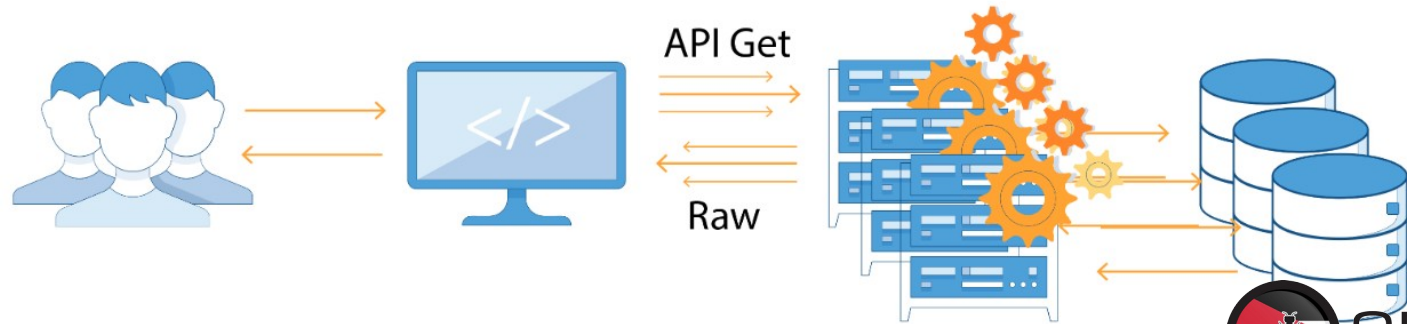
- Globally recognized by developers as the first step towards more secure coding.
- The most critical security risks to web applications.
- Updated every 2-3 years from 2003 to 2017.
- In 2019, OWASP started an effort to create a version of their Top 10 dedicated specifically to API security. The first OWASP API Security Top 10 list was released on 31 December 2019.
- Project link :
 - <https://owasp.org/www-project-api-security/>

Web application security vs API security

Traditional Application

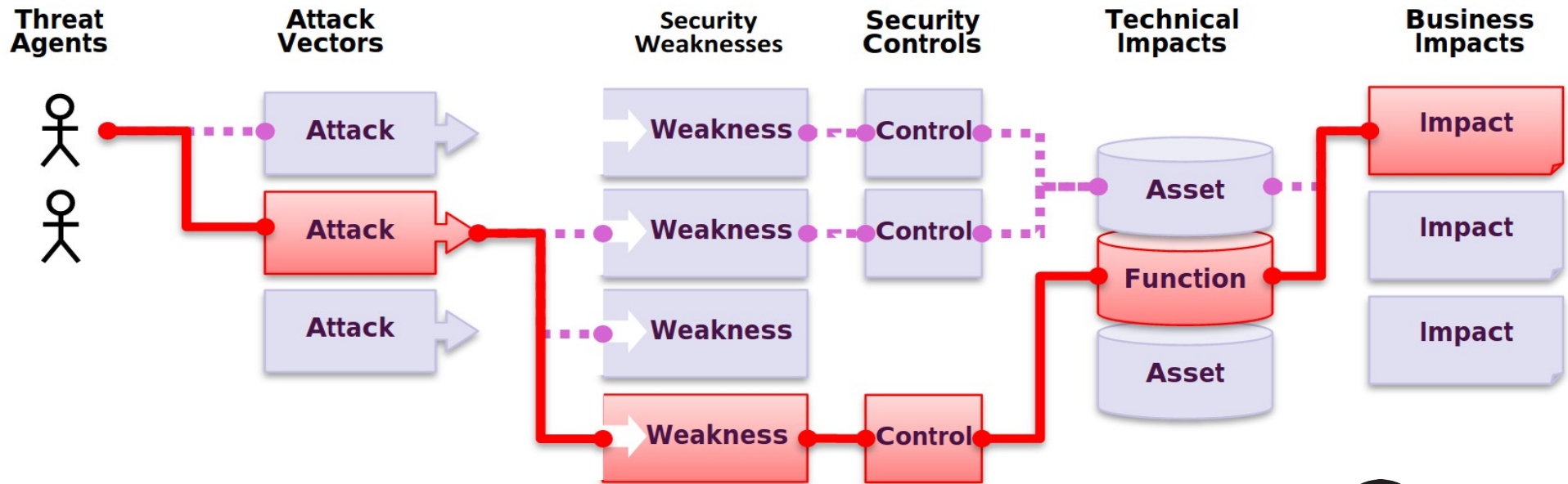


Modern Application



OWASP
Open Web Application
Security Project

What Are Application Risks?



Web Application Risks

- The OWASP API Top 10 focuses on identifying the most serious web application security risks for a broad array of organizations. For each of these risks, we provide generic information about likelihood and technical impact using the following simple ratings scheme, which is based on the OWASP Risk Rating Methodology :
 - https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology

Threat Agents	Exploitability	Weakness Prevalence	Weakness Detectability	Technical Impact	Business Impacts
API Specific	Easy: 3	Widespread 3	Easy 3	Severe 3	Business Specific
	Average: 2	Common 2	Average 2	Moderate 2	
	Difficult: 1	Difficult 1	Difficult 1	Minor 1	



OWASP
Open Web Application
Security Project

OWASP API Top 10 2019

- A1 : Broken object level authorization
- A2 : Broken authentication
- A3 : Excessive data exposure
- A4 : Lack of resources and rate limiting
- A5 : Broken function level authorization
- A6 : Mass assignment
- A7 : Security misconfiguration
- A8 : Injection
- A9 : Improper assets management
- A10 : Insufficient logging and monitoring

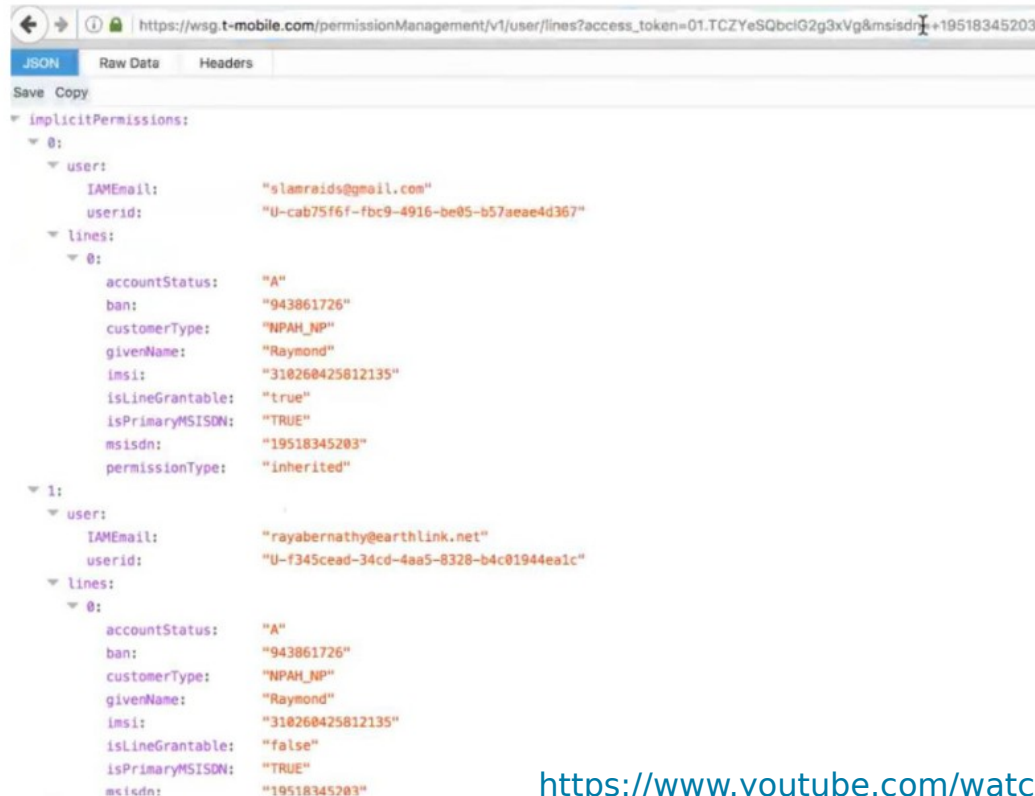
A1 : Broken object level authorization

- APIs consume a lot of object IDs by design:
 - URL params (/api/users/717) / Query Params (/download_file?id=111)
 - Body params / HTTP Headers (user-id:717)
- Known also as:
 - IDOR
 - Forceful Browsing
 - Parameter Tampering
 - Broken Authorization

Continue..



T-Mobile API Breach (2017)



- API behind a web portal
- Phone numbers as IDs
- Was exploited in the wild (e.g. “SIM swap”)

https://www.youtube.com/watch?v=3_gd3a077RU

Prevention

- Implement authorization checks with user policies and hierarchy.
- Do not rely on IDs that the client sends. Use IDs stored in the session object instead.
- Check authorization for each client request to access database.
- Use random IDs that cannot be guessed (UUIDs).

A2 : Broken authentication

- Unprotected APIs that are considered “internal”
- Weak authentication that does not follow industry best practices
- Weak API keys that are not rotated
- Passwords that are weak, plain text, encrypted, poorly hashed, shared, or default passwords
- Authentication susceptible to brute force attacks and credential stuffing
- Credentials and keys included in URLs
- Lack of access token validation (including JWT validation)
- Unsigned or weakly signed non-expiring JWTs

Balboa hot tubs (2018)



<https://www.youtube.com/watch?v=SyL3zG1FtKg>

<https://www.pentestpartners.com/security-blog/hackers-in-hot-water-pwning-smart-hot-tubs-yes-really/>

- The APIs are “protected” with a shared hardcoded password.
- The mobile app controlling the hot tub is invoking the hot tub’s API over the internet.
- Each hot tub comes with an unprotected WiFi hotspot.
- The APIs use the WiFi hotspot ID as the device identifier.
- Attackers can use WiFi hotspot directories to locate all Balboa hot tubs. This gives them both the ID and the location of a hot tub. After that, they can invoke the APIs of the hot tub, know when it is in use, and take over control.

Prevention

- Check all possible ways to authenticate to all APIs.
- APIs for password reset and one-time links also allow users to authenticate, and should be protected just as rigorously.
- Use standard authentication, token generation, password storage, and multi-factor authentication (MFA).
- Use short-lived access tokens.
- Authenticate your apps (so you know who is talking to you).
- Use stricter rate-limiting for authentication, and implement logout policies and weak password checks.

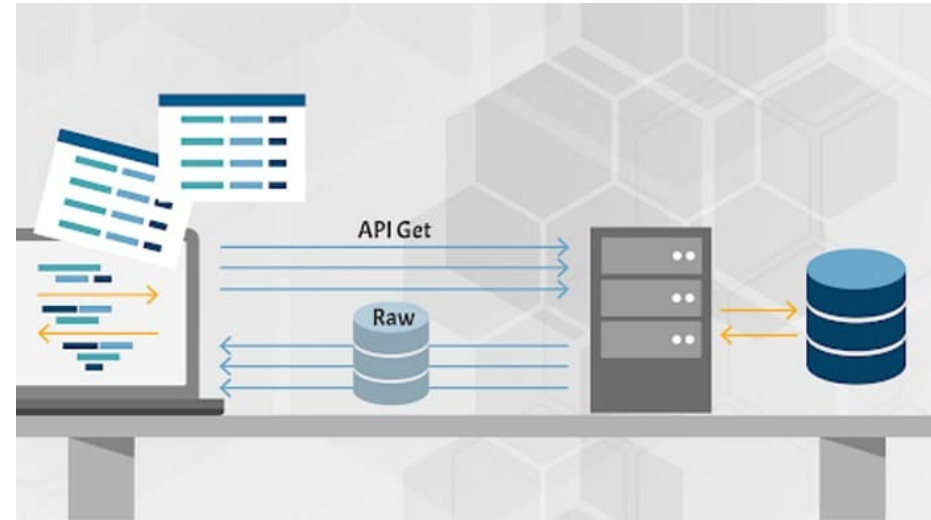


OWASP

Open Web Application
Security Project

A3 : Excessive data exposure

- The API returns full data objects as they are stored in the backend database.
- The client application filters the responses and only shows the data that the users really need to see.
- Attackers call the API directly and get also the sensitive data that the UI would filter out.



Uber account takeover (2019)

- Error message leaking user UUID

<https://vimeo.com/359252275>

Request

```
POST /p3/fleet-manager/_rpc?rpc=addDriverV2 HTTP/1.1
Host: partners.uber.com
{"nationalPhoneNumber":"99999xxxxx","countryCode":"1"}
```

Response

```
{
  "status":"failure",
  "data": {
    "code":1009,
    "message":"Driver '47d063f8-0xx5e-xxxxx-b01a-xxxx' not found"
  }
}
```



OWASP
Open Web Application
Security Project

Continue..

- Make a request with the UUID

Request

```
POST /marketplace/_rpc?rpc=getConsentScreenDetails HTTP/1.1
Host: bonjour.uber.com
Connection: close
Content-Length: 67
Accept: application/json
Origin: [https://bonjour.uber.com](https://bonjour.uber.com)
x-csrf-token: xxxx
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_3) AppleWebKit/537.36
DNT: 1
Content-Type: application/json
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: xxxxx
{"language":"en","userId":"xxxx-776-4xxxx1bd-861a-837xxx604ce"}
```

Continue..

- Receive tons of user data including mobile app authentication token

Response
(173 lines!!!)

```
{
  "status": "success",
  "data": {
    "language": "en",
    "userUuid": "xxxxxx1e"
  },
  "currentUser": {
    "uuid": "cxxxxxc5f7371e",
    "firstname": "Maxxxx",
    "lastname": "XXXX",
    "role": "PARTNER",
    "languageId": 1,
    "countryId": 77,
    "mobile": null,
    "mobileToken": 1234,
    "mobileCountryId": 77,
    "mobileCountryCode": "+91",
    "hasAmbiguousMobileCountry": false,
    "lastConfirmedMobileCountryId": 77,
    "email": "xxxx@gmail.com",
    "emailToken": "xxxxxxx",
    "hasConfirmedMobile": "no",
    "hasOptedInSmsMarketing": false,
    "hasConfirmedEmail": true,
    "gratuity": 0.3,
    "nickname": "abc@gmail.com",
    "location": "00000",
    "banned": false,
    "cardio": false,
    "token": "b8038ec4143bb4xxxxxx72d",
    "fraudScore": 0,
    "inviterUuid": null,
    "pictureUrl": "xxxxx.jpeg",
    "recentFareSplitterUuids": [
      "xxx"
    ],
    "lastSelectedPaymentProfileUuid": "xxxxxx",
    "lastSelectedPaymentProfileGoogleWalletUuid": null,
    "inviteCode": {
      "promotionCodeId": xxxxx,
      "promotionCodeUuid": "xxxx",
      "promotionCode": "manishas105",
      "createdAt": {
        "type": "Buffer",
        "data": [0, 0, 1, 76, 2, 21, 215, 101]
      },
      "updatedAt": {
        "type": "Buffer",
        "data": [0, 0, 1, 76, 65, 211, 61, 9]
      }
    },
    "driverInfo": {
      "contactInfo": "999999999xx",
      "contactInfoCountryCode": "+91",
      "driverLicense": "None",
      "firstDriverTripUuid": null,
      "iphone": null,
      "partnerUserUuid": "xxxxxxx",
      "receiveSms": true,
      "twilioNumber": null,
      "twilioNumberFormatted": null,
      "cityKnowledgeScore": 0,
      "createdAt": {
        "type": "Buffer",
        "data": [0, 0, 1, 84, 21, 124, 80, 52]
      },
      "updatedAt": {
        "type": "Buffer",
        "data": [0, 0, 1, 86, 152, 77, 41, 77]
      },
      "deletedAt": null,
      "driverStatus": "APPLIED",
      "driverFlowType": "UBERX",
      "statusLocks": null,
      "contactInfoCountryIso2Code": "KR",
      "driverEngagement": null,
      "courierEngagement": null
    },
    "partnerInfo": {
      "address": "Nxxxxxxx",
      "territoryUuid": "xxxxxx",
      "company": "None",
      "address2": "None",
      "cityId": 130,
      "cityName": "None",
      "firstPartnerTripUuid": null,
      "preferredCollectionPaymentProfileUuid": null,
      "phone": "",
      "phoneCountryCode": "+91",
      "state": "None",
      "vatNumber": "None",
      "zipCode": "None"
    }
  }
}
```



Prevention

- Never rely on the client to filter data!
- Review all API responses and adapt them to match what the API consumers really need.
- Carefully define schemas for all the API responses.
- Do not forget about error responses, define proper schemas as well.
- Identify all the sensitive data or Personally Identifiable Information (PII), and justify its use.
- Enforce response checks to prevent accidental leaks of data or exceptions.

A4: Lack of Resources & Rate Limiting

- Attackers overload the API by sending more requests than it can handle.
- Attackers send requests at a rate exceeding the API's processing speed, clogging it up.
- The size of the requests or some fields in them exceed what the API can process.
- “Zip bombs”, archive files that have been designed so that unpacking them takes excessive amount of resources and overloads the API.



Instagram account takeover (2019)

- Used password reset API
- Passcode expires in 10 minutes
- Rate limiting 200 / minute / IP
- 5000 cloud VMs can enumerate all 6-digit codes
- AWS/GCP cost ~ \$150

<https://www.youtube.com/watch?v=4O9FjTMIHUM>

Requesting passcode

POST /api/v1/users/lookup/

Host: i.instagram.com

q=mobile_number&
device_id=android-device-id

Verify passcode

POST /api/v1/accounts
/account_recovery_code_verify/

Host: i.instagram.com

recover_code=123456&
device_id=android-device-id



OWASP
Open Web Application
Security Project

Prevention

- Define proper rate limiting.
- Limit payload sizes.
- Tailor the rate limiting to be match what API methods, clients, or addresses need or should be allowed to get.
- Add checks on compression ratios.
- Define limits for container resources.

A5: Broken Function Level Authorization

- The API relies on the client to use user level or admin level APIs as appropriate. Attackers figure out the “hidden” admin API methods and invoke them directly.
 - Some administrative functions are exposed as APIs.
 - Non-privileged users can access these functions without authorization if they know how.
 - Can be a matter of knowing the URL, or using a different verb or a parameter:
 - `/api/users/v1/user/myinfo`
 - `/api/admins/v1/users/all`



Gator kids smartwatches (2019)

- Simple request of User[Grade] value from 1 to 0 enabled management of all smartwatches

<https://www.pentestpartners.com/security-blog/gps-watch-issues-again/>

Request

Raw Params Headers Hex

POST request to /web/index.php

Type	Name	Value
URL	r	secured/user/profile
Cookie	_csrf	e432ab101109a4936950ca7329145a5fe444c4fe3
Cookie	PHPSESSID	dduvqj68euk6b930jasq1oqmu4
Body	_csrf	N09fc2szcHdxCTheD2slA00kGDAFajs6fR8oBiN
Body	User[recid]	7e837ebd-18b5-11e9-a49c-0a6fca88bf80
Body	User[Grade]	1
Body	User[companyId]	007BA0BE-7168-43D3-8A41-C502FC3F4DCF
Body	User[NickName]	egw2
Body	User[BossId]	05CD69A2-4DC0-42EB-8351-401983D1
Body	User[XzAddress]	
Body	User[LinkMan]	
Body	User[Contact]	
Body	User[Fax]	
Body	User[Email]	
Body	User[dateformat]	yyyy-MM-dd
Body	User[datetimeformat]	yyyy-MM-dd



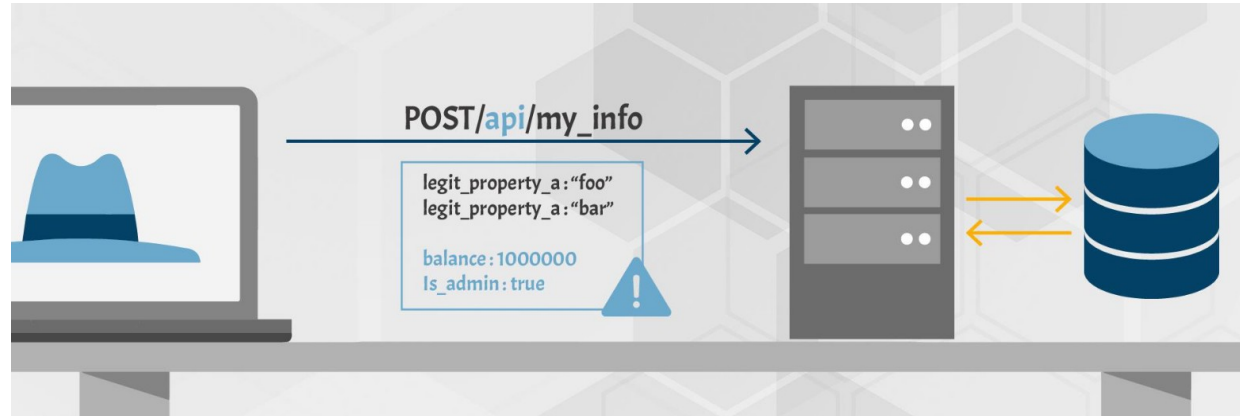
OWASP
Open Web Application
Security Project

Prevention

- Do not rely on the client to enforce admin access.
- Deny all access by default.
- Only allow operations to users belonging to the appropriate group or role.
- Properly design and test authorization.

A6: Mass Assignment

- The API takes data that client provides and stores it without proper filtering for whitelisted properties.
- The API works with the data structures without proper filtering.
- Received payload is blindly transformed into an object and stored.
- Attackers can guess the fields by looking at the GET request data.



Harbor (2019)

```
type User struct {
    UserID int    `orm:"pk;auto;column(user_id)" json:"user_id"`
    Username string `orm:"column(username)" json:"username"`
    Email string  `orm:"column(email)" json:"email"`
    Password string `orm:"column(password)" json:"password"`
    Realname string `orm:"column(realname)" json:"realname"`
    Comment string `orm:"column(comment)" json:"comment"`
    Deleted bool    `orm:"column(deleted)" json:"deleted"`
    RoleName string `orm:"- " json:"role_name"`
    // if this field is named as "RoleID", beego orm can not map role_id
    // to it.
    Role int `orm:"- " json:"role_id"`
    // RoleList []Role `json:"role_list"`
    HasAdminRole bool `orm:"column(sysadmin_flag)" json:"has_admin_role"`
    ResetUUID string `orm:"column(reset_uuid)" json:"reset_uuid"`
    Salt string `orm:"column(salt)" json:"- "`
    CreationTime time.Time `orm:"column(creation_time);auto_now_add" json:"creation_time"`
    UpdateTime time.Time `orm:"column(update_time);auto_now" json:"update_time"`
    GroupIDs []int `orm:"- " json:"- "`
    OIDCUserMeta *OIDCUser `orm:"- " json:"oidc_user_meta,omitempty"`
}
```

- Attacker can make himself admin.

```
POST /api/users
{"username":"test","email":"test123@gmai.com",
"realname":"no
name","password":"Password1\u0021","comment":
null,
"has_admin_role":true}
```

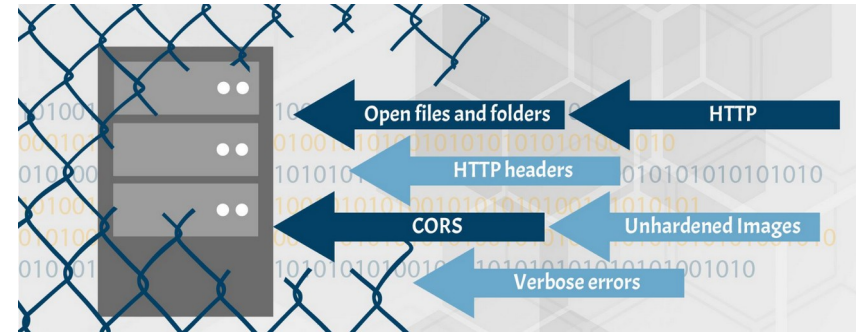
<https://www.youtube.com/watch?v=LBgIKqdfF1k>

Prevention

- Do not automatically bind incoming data and internal objects.
- Explicitly define all the parameters and payloads you are expecting.
- Use the readOnly property set to true in object schemas for all properties that can be retrieved through APIs but should never be modified.
- Precisely define the schemas, types, and patterns you will accept in requests at design time and enforce them at runtime.

A7: Security Misconfiguration

- Poor configuration of the API servers allows attackers to exploit them.
 - Unpatched systems
 - Unprotected files and directories
 - Unhardened images
 - Missing, outdated, or misconfigured TLS
 - Exposed storage or server management panels
 - Missing CORS policy or security headers
 - Error messages with stack traces
 - Unnecessary features enabled



Equifax

- In 2017, Equifax was breached and private information of 147 million people got exposed. The breach started with unpatched Apache Struts exploit. The system was not enforcing formats on incoming API calls and specifically crafted Content-Type header allowed attackers to get in.

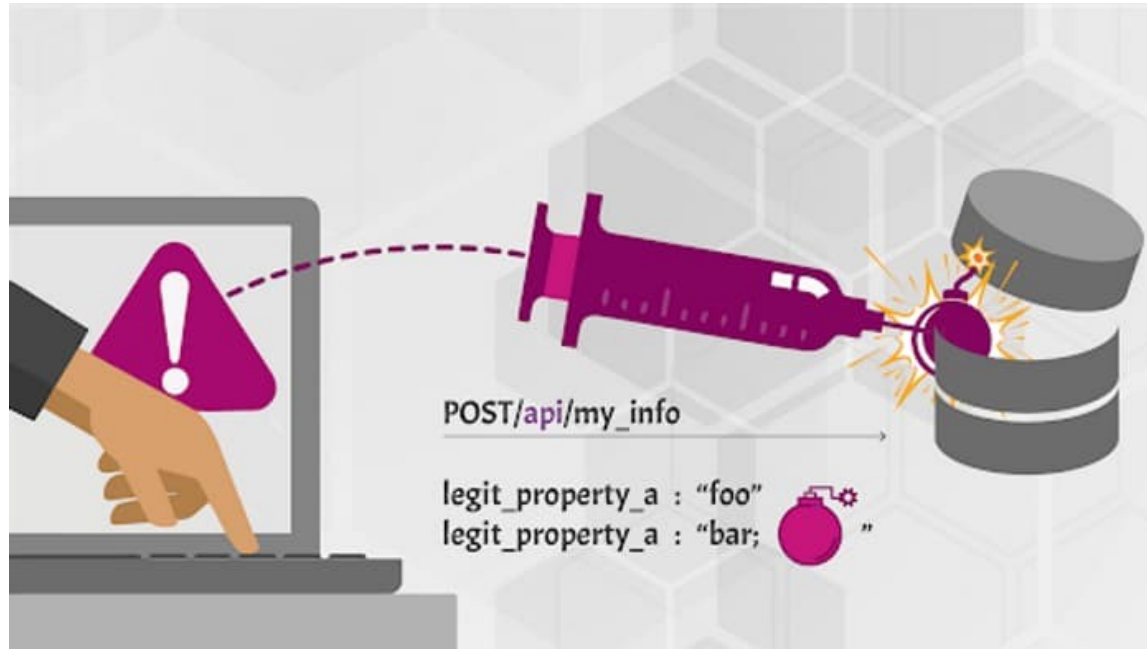
<https://republicans-oversight.house.gov/wp-content/uploads/2018/12/Equifax-Report.pdf>

Prevention

- Establish repeatable hardening and patching processes.
- Automate locating configuration flaws.
- Disable unnecessary features.
- Restrict administrative access.
- Define and enforce all outputs, including errors.

A8: Injection

- Attackers construct API calls that include SQL, NoSQL, LDAP, OS, or other commands that the API or the backend behind it blindly executes.



Samsung SmartThings Hub (2018)

- SmartThings Hub is able to communicate with cameras
- credentials API can be used to set credentials for remote server authentication
- Data is saved in SQLite database
- JSON keys with ; allowed to execute arbitrary queries

```
# using curl from inside the hub, but the same request could be sent using a SmartApp
$ sInj='","_id=0 where 1=2;insert into camera values (123,replace(substr(quote(zeroblob((10000 + 1) /
2)), 3, 10000), "\\0\\", "\\A\\"),1,1,1,1,1,1,1,1,1,1,1,1,1,1);--":"'
$ curl -X POST 'http://127.0.0.1:3000/credentials' -d
'{"s3":{"accessKey":"","secretKey":"","directory":"","region":"","bucket":"","sessionToken":"'${sInj}'"},'
videoHostUrl':'127.0.0.1/'}"
$ curl -X DELETE "http://127.0.0.1:3000/cameras/123"
```

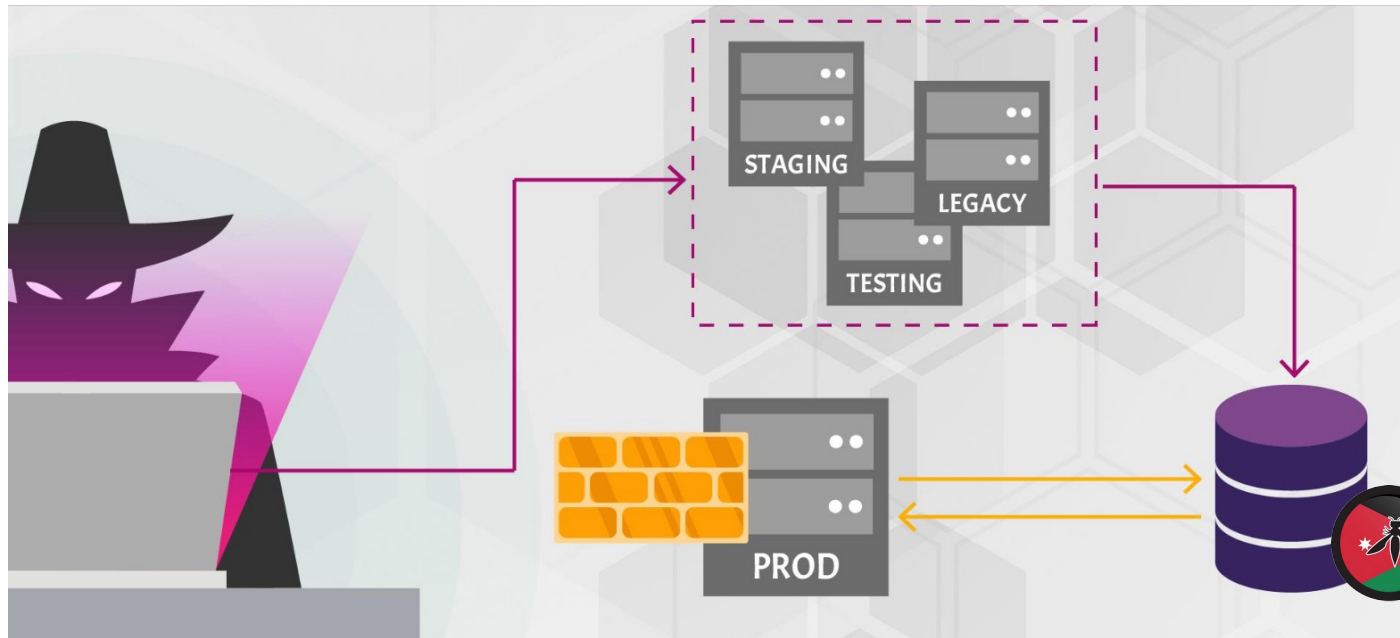
https://talosintelligence.com/vulnerability_reports/TALOS-2018-0556/

Prevention

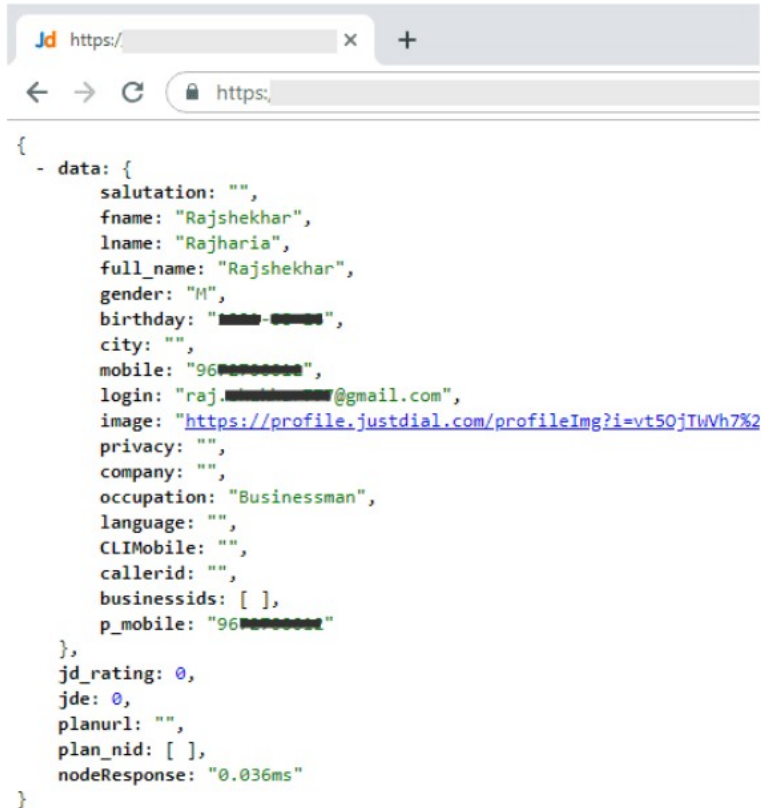
- Never trust your API consumers, even if internal
- Strictly define all input data: schemas, types, string patterns -and enforce them at runtime
- Validate, filter, sanitize all incoming data
- Define, limit, and enforce API outputs to prevent data leaks

A9: Improper Assets Management

- Attackers find non-production versions of the API (for example, staging, testing, beta, or earlier versions) that are not as well protected as the production API, and use those to launch their attacks.



JustDial(2019)



```
{
  - data: {
    salutation: "",
    fname: "Rajshekhar",
    lname: "Rajharia",
    full_name: "Rajshekhar",
    gender: "M",
    birthday: "1990-08-08",
    city: "",
    mobile: "9600000000",
    login: "raj.000000@gmail.com",
    image: "https://profile.justdial.com/profileImg?i=vt50jTWVh7%2
    privacy: "",
    company: "",
    occupation: "Businessman",
    language: "",
    CLIMobile: "",
    callerid: "",
    businessids: [ ],
    p_mobile: "9600000000"
  },
  jd_rating: 0,
  jde: 0,
  planurl: "",
  plan_nid: [ ],
  nodeResponse: "0.036ms"
}
```

- India's largest local search service
- Had old unused unprotected API
- It was connected to live database
- Was leaking 100 mlnusers' data

Tchap

- Tchap is a messaging app that the French government released for internal use. It was hailed as a more secure replacement for Telegram and WhatsApp.
 - The sign-up API had an email address parameter that didn't validate the input format.
 - A security researcher, Elliot Alderson, submitted `fs0c131y@protonmail.com@elysee.fr` as the address.
 - The code simply verified that the address ended with `@elysee.fr`, which is a government email domain.
 - The check was successful, and Tchap sent a verification email that got delivered to `fs0c131y@protonmail.com` belonging to the attacker.
 - The attacker could click the confirmation link, get in, and be able to get into the internal government chat rooms.

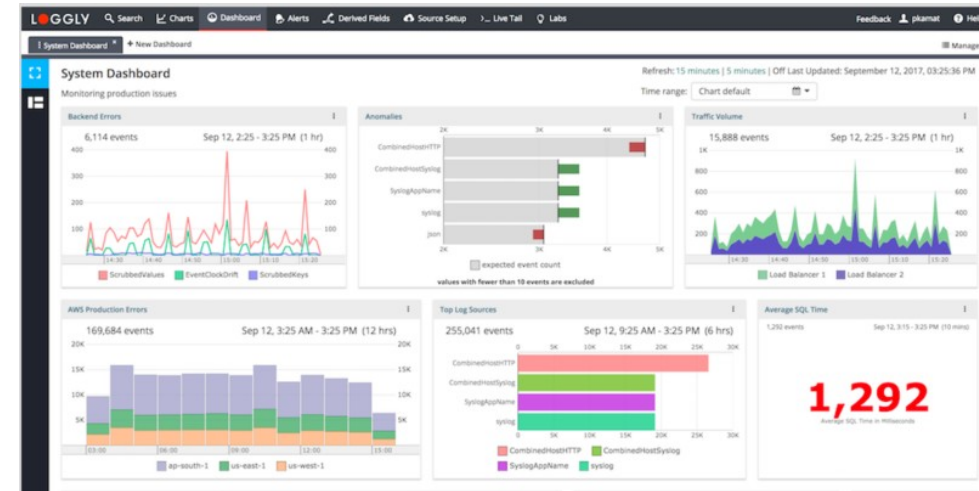
Prevention

- Keep an up-to-date inventory all API hosts.
- Limit access to anything that should not be public.
- Limit access to production data, and segregate access to production and non-production data.
- Implement additional external controls, such as API firewalls.
- Properly retire old versions of APIs or backport security fixes to them.
- Implement strict authentication, redirects, CORS, and so forth.



A10: Insufficient Logging & Monitoring

- You can't react to attacks that you don't know about.
- Logs are important for:
 - Detecting incidents
 - Understanding what happened
 - Proving who did something



7-Eleven Japan: 7Pay

- Payment app
- Password reset allowed email change if personal details were supplied
- Attackers used API to try combinations of info for various Japanese citizens
- \$510K was stolen from 900 customers
- The company only noticed after too many users complained

× 7 i Dでログイン

オムニ7会員から7 i D会員に名称が変更になりました。会員ID・パスワードはそのままご利用いただけます。

7 i D (メールアドレスまたは任意の文字列)

パスワード

半角英数字2種類以上組み合わせ(8文字以上)

[7 i Dをお忘れの方](#)

[パスワードをお忘れの方](#)

[他のサイトIDでログイン](#)

 ログイン

 ログイン

OWASP
Open Web Application
Security Project

Prevention

- Log failed attempts, denied access, input validation failures, any failures in security policy checks
- Ensure that logs are formatted to be consumable by other tools
- Protect logs as highly sensitive
- Include enough detail to identify attackers
- Avoid having sensitive data in logs -If you need the information for debugging purposes, redact it partially.
- Integrate with SIEMs and other dashboards, monitoring, alerting tools





Any Questions?