OWASP TBILISI

OWASP API Security Top 10

The Ten Most Critical API Security Risks

 /www-chapter-tbilisi/

 /OWASP-Tbilisi-Chapter

 /OWASP.Tbilisi

API Usage Statistics and Predictions

“83% of all web traffic is now API call traffic.”
- Akamai, State of the Internet 2018

“By 2021, exposed APIs will form a larger surface area for attacks than the UI in 90% of web-enabled applications.”
- Gartner, API Strategy Maturity Model, October 2019

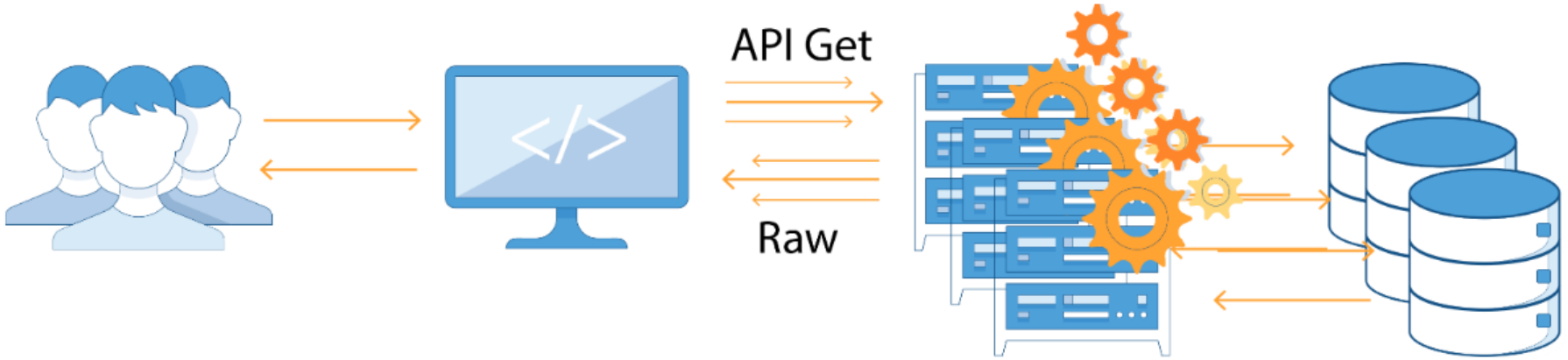
“By 2022, APIs will become the #1 attack vector.”
- Gartner, How to Build an Effective API Security Strategy

OWASP TBILISI

Traditional vs. Modern

Traditional Application

Modern Application



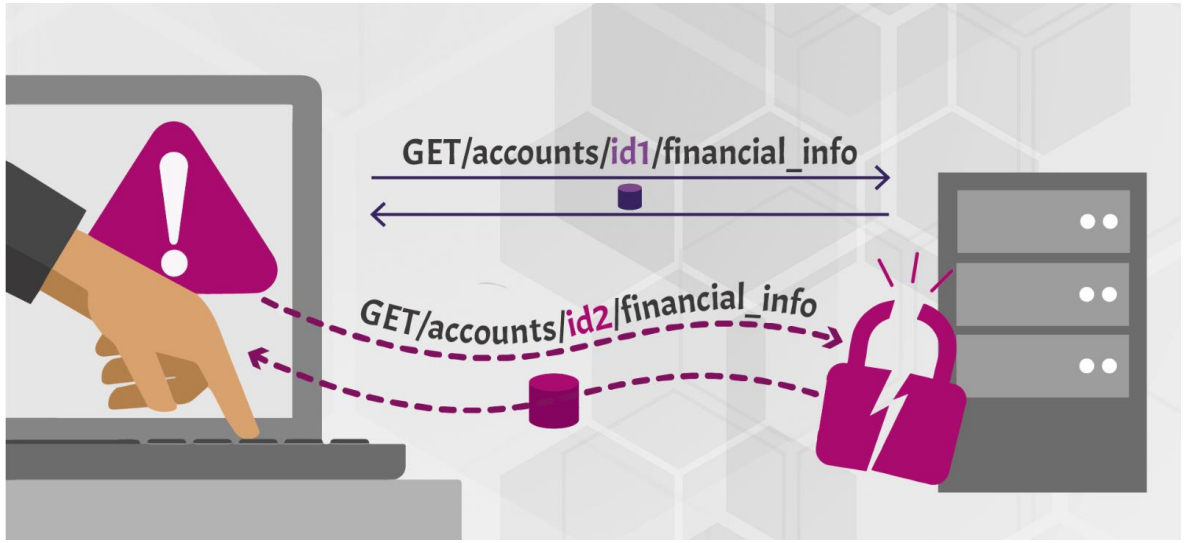
OWASP TBILISI

OWASP API Security Top 10 2019

API1:2019 Broken Object Level Authorization
API2:2019 Broken User Authentication
API3:2019 Excessive Data Exposure
API4:2019 Lack of Resources & Rate Limiting
API5:2019 Broken Function Level Authorization
API6:2019 Mass Assignment
API7:2019 Security Misconfiguration
API8:2019 Injection
API9:2019 Improper Assets Management
API10:2019 Insufficient Logging & Monitoring

OWASP TBILISI

API1:2019 - Broken Object Level Authorization



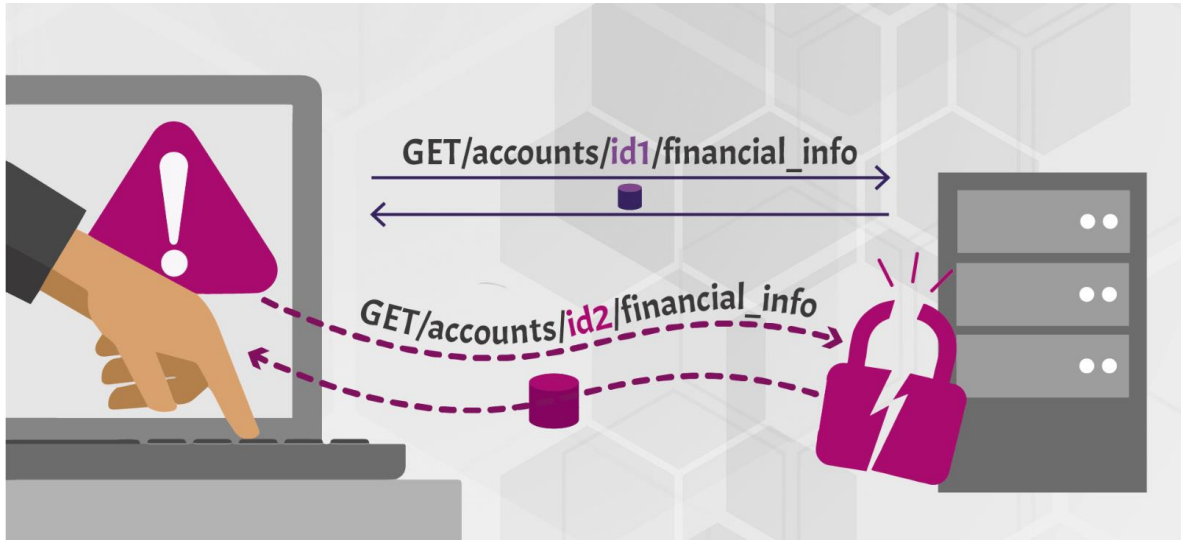
Attacker substitutes ID of their resource in API call with an ID of a resource belonging to another user. Lack of proper authorization checks allows access. This attack is also known as **IDOR** (Insecure Direct Object Reference).

USE CASES

- API call parameters use IDs of resourced accessed by the API:
  /api/**shop1**/financial_details
- Attackers replace the IDs of their resources with different ones, which they guessed:
  /api/**shop2**/financial_details
- The API does not check permissions and lets the call through.
- Problem is aggravated if IDs can be enumerated:
  /api/**123**/financial_details

OWASP TBILISI

Mitigation of API1:2019



Attacker substitutes ID of their resource in API call with an ID of a resource belonging to another user. Lack of proper authorization checks allows access. This attack is also known as **IDOR** (Insecure Direct Object Reference).

HOW TO PREVENT

- Implement authorization checks with user policies and hierarchy.
- Don’t rely on IDs sent from client. Use IDs stored in the session object instead.
- Check authorization each time there is a client request to access database.
- Use random non-guessable IDs (UUIDs).

OWASP TBILISI

API2:2019 - Broken User Authentication

Poorly implemented API authentication allowing attackers to assume **other users identities**.

USE CASES

- Unprotected APIs that are considered “internal”.
- Weak authentication not following industry best practices.
- Weak, not rotating API keys.
- Weak, plain text, encrypted, poorly hashed, shared/default passwords.
- Susceptible to brute force attacks and credential stuffing.
- Credentials and keys in URL.
- Lack of access token validation (including JWT validation).
- Unsigned, weakly signed, non-expiring JWTs.

OWASP TBILISI

Mitigation of API2:2019

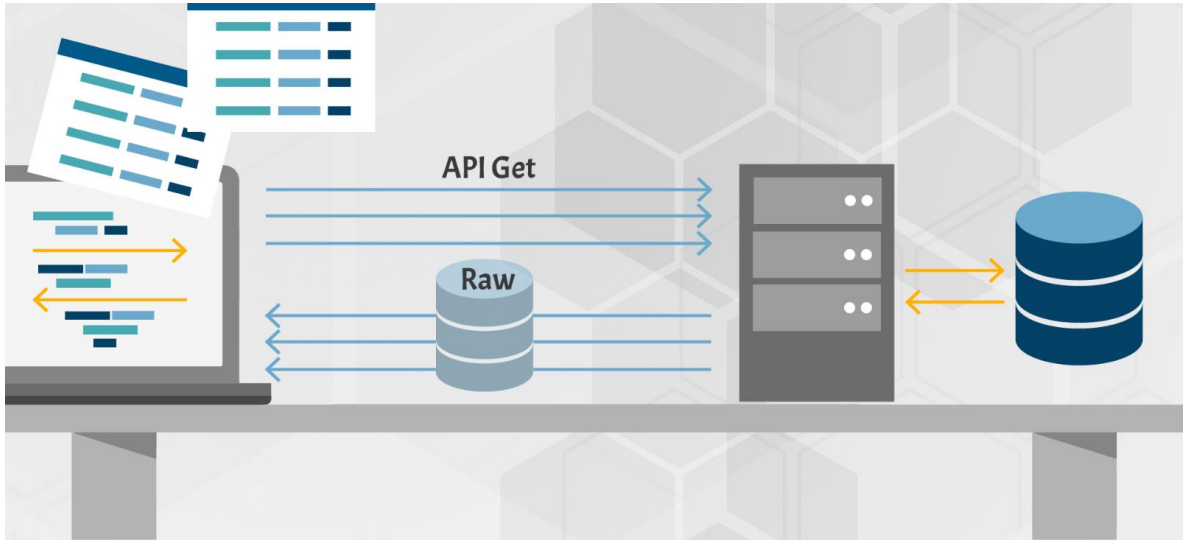
Poorly implemented API authentication allowing attackers to assume **other users identities**.

HOW TO PREVENT

- Check all possible ways to authenticate to all APIs.
- Password reset APIs and one-time links also allow users to get authenticated and should be protected just as seriously.
- Use standard authentication, token generation, password storage, Multifactor authentication.
- Use short-lived access tokens.
- Use stricter rate-limiting for authentication, implement lockout policies and weak password checks.

OWASP TBILISI

API3:2019 - Excessive Data Exposure



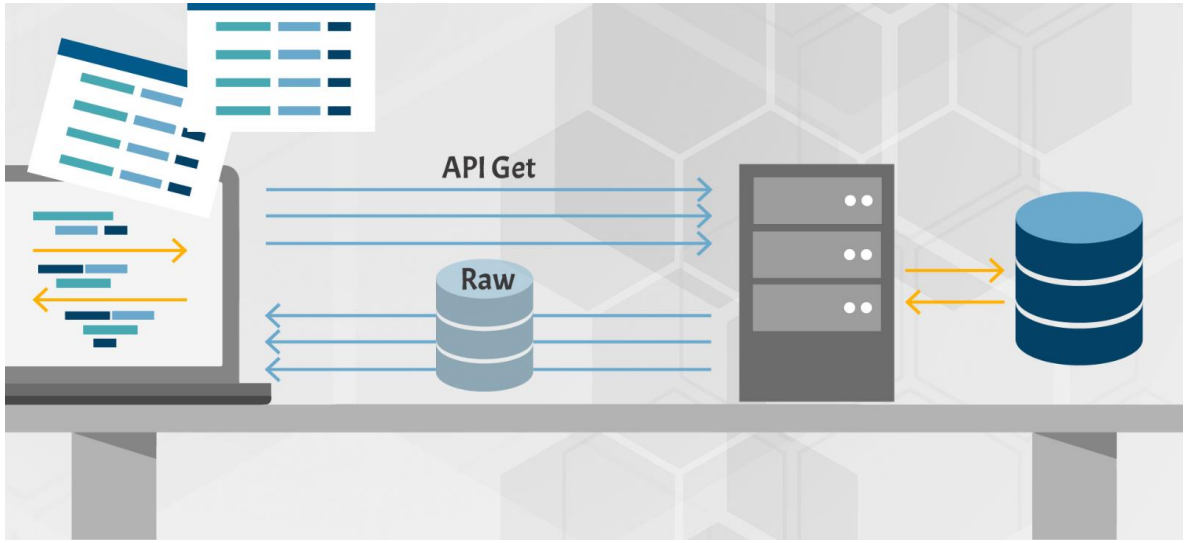
API exposing a **lot more data** than the client legitimately needs, relying on the client to do the filtering. Attacker goes directly to the API and has it all.

USE CASES

- APIs return full data objects as they are stored by the database.
- Client application shows only the data that user needs to see.
- Attacker calls the API directly and gets sensitive data.

OWASP TBILISI

Mitigation of API3:2019



API exposing a **lot more data** than the client legitimately needs, relying on the client to do the filtering. Attacker goes directly to the API and has it all.

HOW TO PREVENT

- Never rely on client to filter data.
- Review all responses and adapt responses to what the API consumers really need.
- Define schemas of all the API responses.
- Don’t forget about error responses.
- Identify all the sensitive or PII info and justify its use.
- Enforce response checks to prevent accidental data and exception leaks.

OWASP TBILISI

API4:2019 - Lack of Resources & Rate Limiting

API is not protected against an excessive amount of calls or payload sizes. Attackers use that for **DoS** and **brute force** attacks.

USE CASES

- Attacker overloading the API
- Excessive rate of requests
- Request or field sizes
- “Zip bombs”

OWASP TBILISI

Mitigation of API4:2019

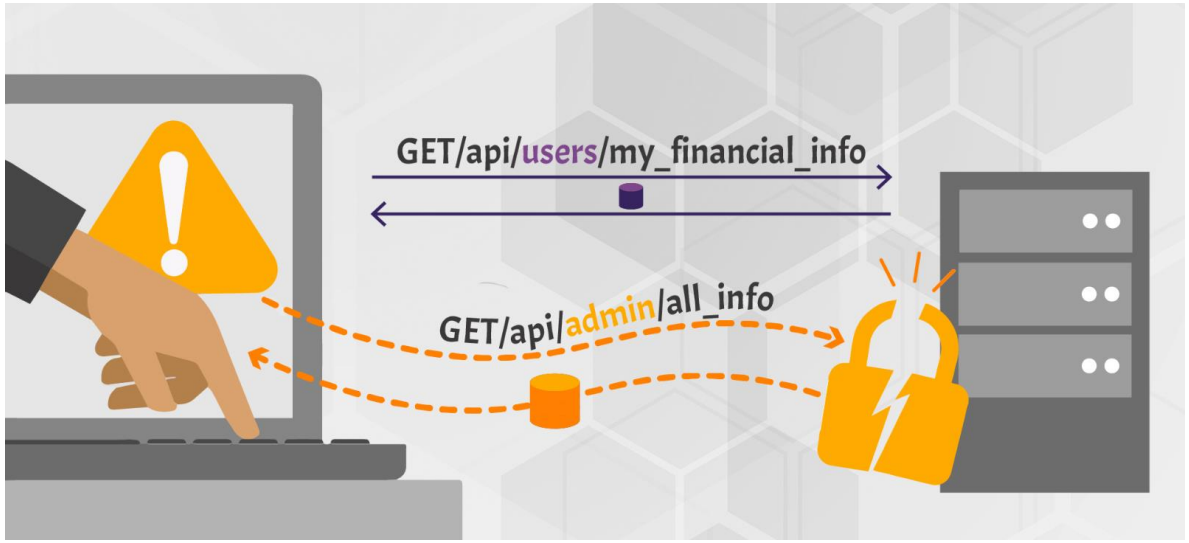
API is not protected against an excessive amount of calls or payload sizes. Attackers use that for **DoS** and **brute force** attacks.

HOW TO PREVENT

- Rate limiting.
- Payload size limits.
- Rate limits specific to API methods, clients, addresses.
- Checks on compression ratios.
- Limits on container resources.
- Check parsers on recursion vulnerabilities.

OWASP TBILISI

API5:2019 - Broken Function Level Authorization



API relies on client to use user level or admin level APIs. Attacker figures out the **"hidden" admin API methods** and invokes them directly.

USE CASES

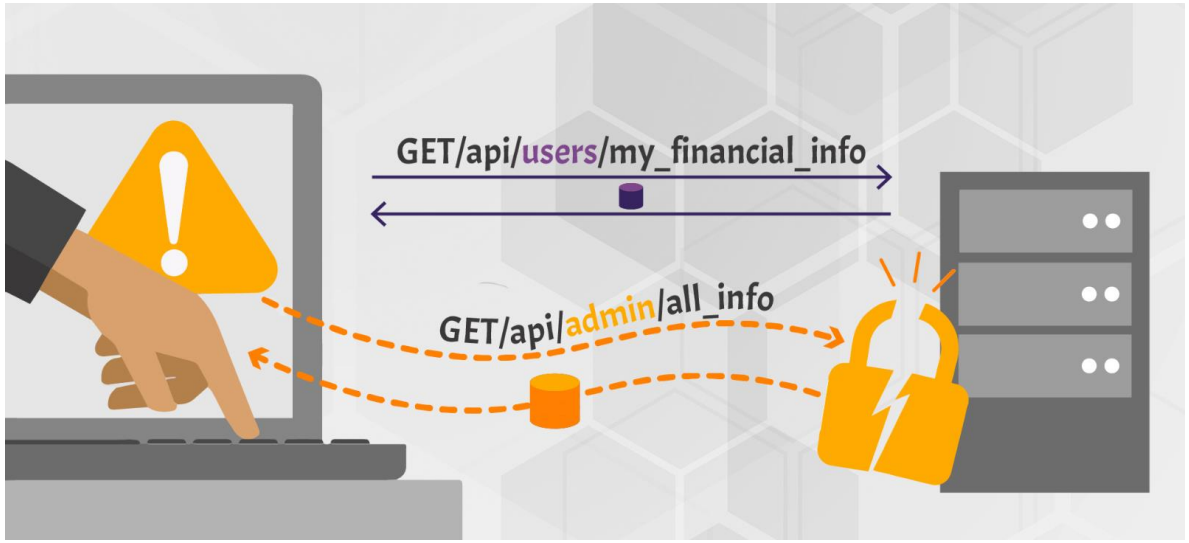
- Some administrative functions are exposed as APIs.
- Non-privileged users can access these functions if they know how.
- Can be a matter of knowing the URL, using a different verb or parameter

  /api/**users**/v1/user/myinfo

  /api/**admins**/v1/users/all

OWASP TBILISI

Mitigation of API5:2019



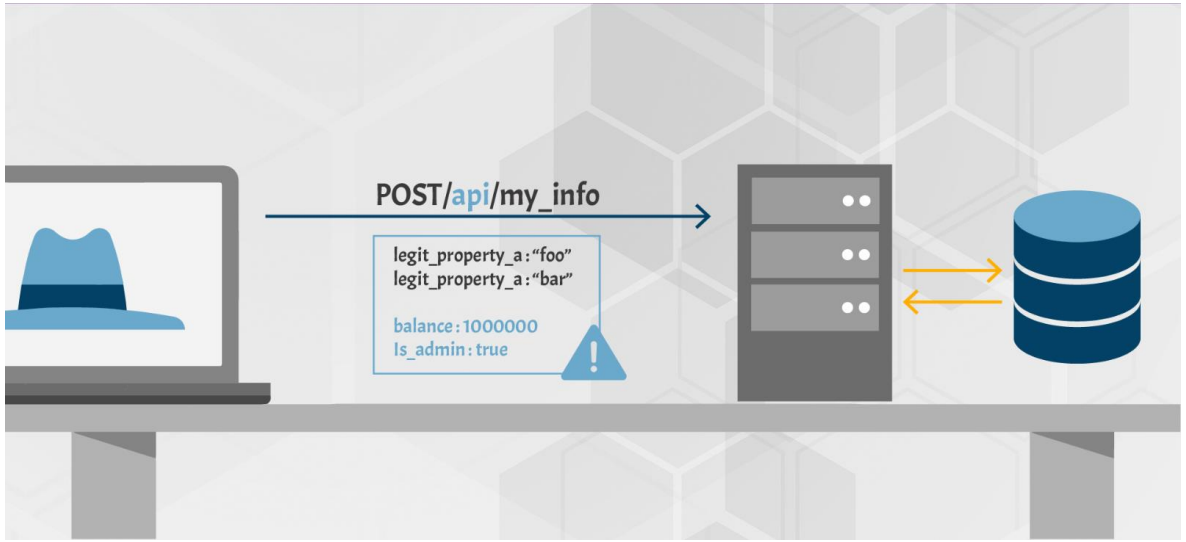
API relies on client to use user level or admin level APIs. Attacker figures out the **“hidden” admin API methods** and invokes them directly.

HOW TO PREVENT

- Don’t rely on app to enforce admin access.
- Deny all access by default.
- Grant access based on specific roles.
- Properly design and test authorization.

OWASP TBILISI

API6:2019 - Mass Assignment



API received payload is **blindly transformed into an object** and **stored**.

USE CASES

- API working with the data structures.
- Received payload is blindly transformed into an object and stored.

  NodeJS:

  ```
  var user = new User(req.body);
  user.save();
  ```

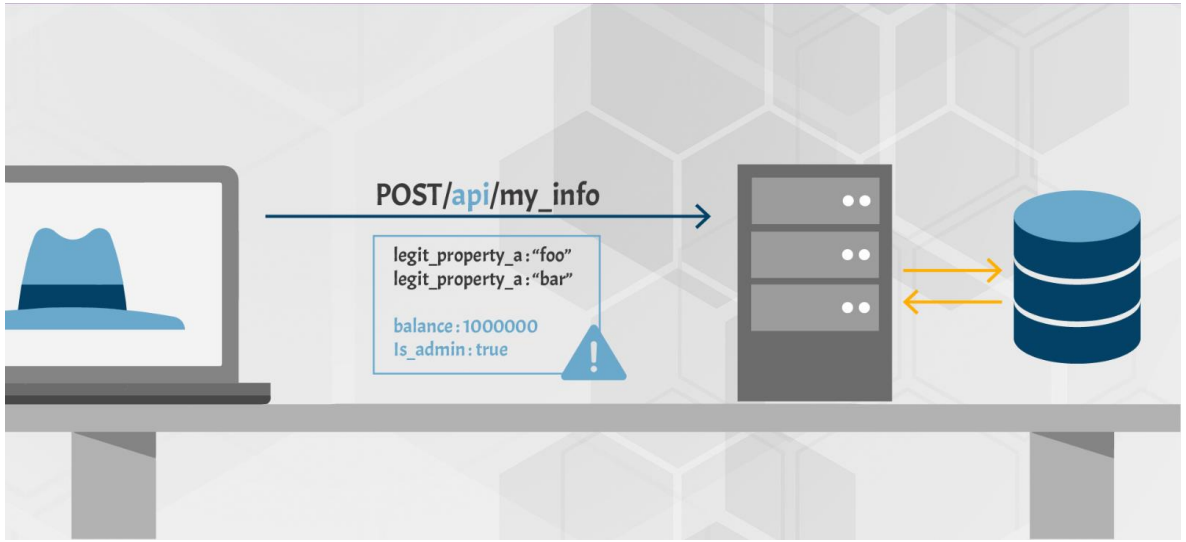
  Rails:

  ```
  @user = User.new(params[:user])
  ```

- Attackers can guess the fields by looking at the GET request data.

OWASP TBILISI

Mitigation of API6:2019



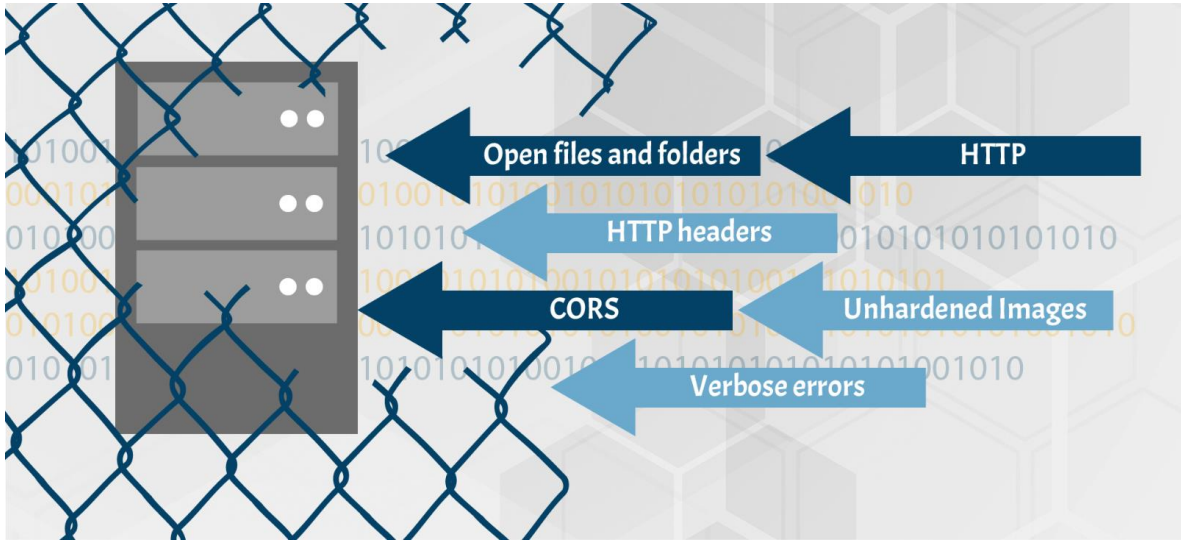
API received payload is **blindly transformed into an object** and **stored**.

HOW TO PREVENT

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

OWASP TBILISI

API7:2019 - Security Misconfiguration



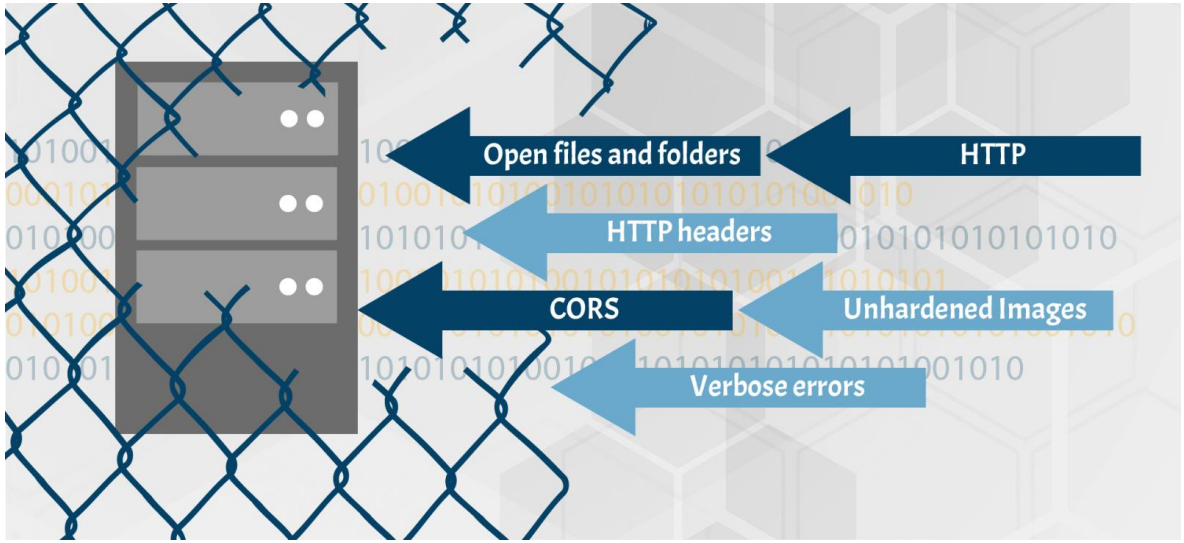
Poor configuration of the API servers allows attackers to exploit them.

USE CASES

- Unpatched systems.
- Unprotected files and directories.
- Unhardened images.
- Missing, outdated, misconfigured TLS.
- Exposed storage or server management panels.
- Missing CORS policy or security headers.
- Error messages with stack traces.
- Unnecessary features enabled.

OWASP TBILISI

Mitigation of API7:2019



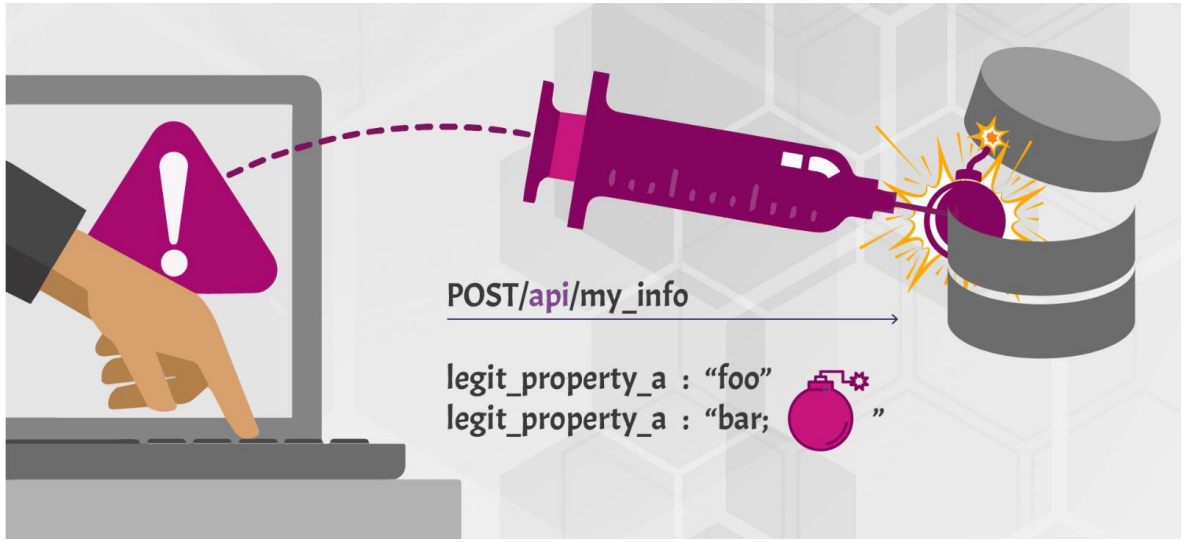
Poor configuration of the API servers allows attackers to exploit them.

HOW TO PREVENT

- Repeatable hardening and patching processes.
- Automated process to locate configuration flaws.
- Disable unnecessary features.
- Restrict administrative access.
- Define and enforce all outputs including errors.

OWASP TBILISI

API8:2019 - Injection



Attacker constructs API calls that include SQL-, NoSQL, LDAP, OS and other **commands** that the API or backend behind it **blindly executes**.

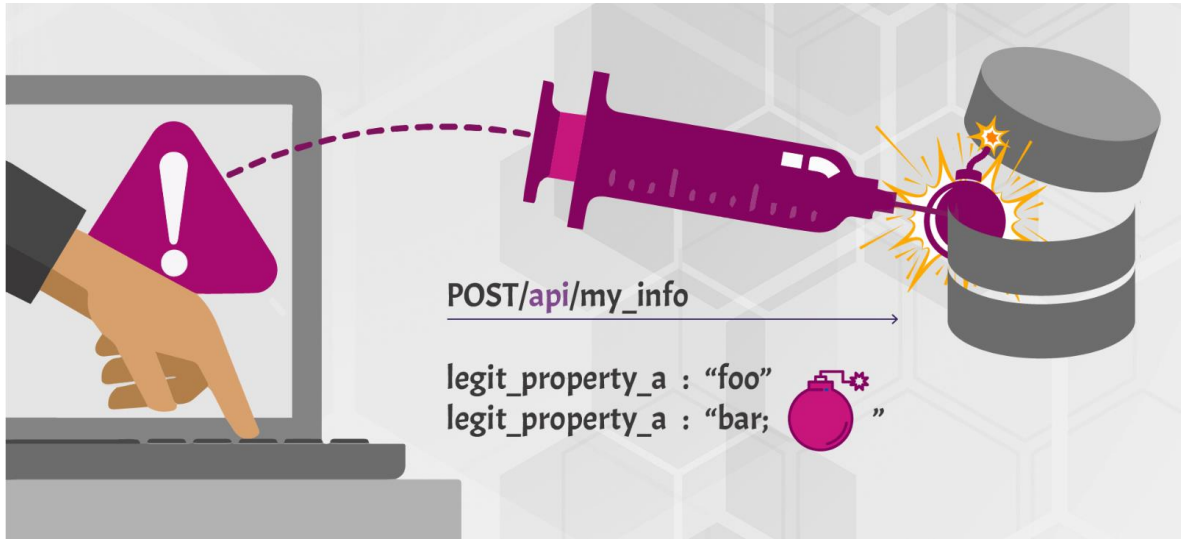
USE CASES

Attackers send **malicious input** to be forwarded to an internal interpreter:

- SQL, NoSQL
- LDAP
- OS commands
- XML parsers
- Object-Relational Mapping (ORM)

OWASP TBILISI

Mitigation of API8:2019



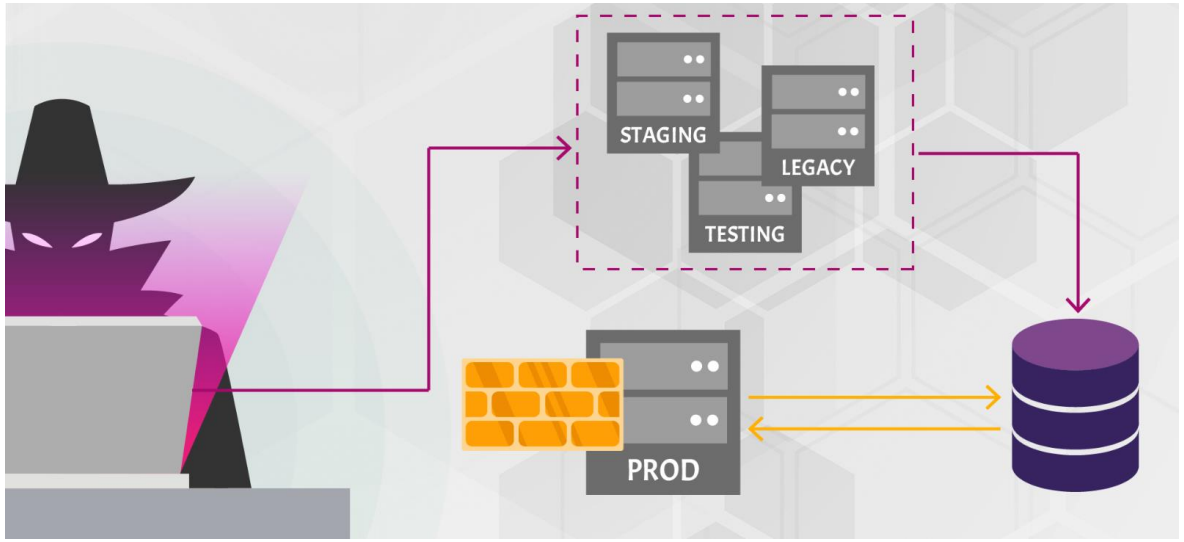
Attacker constructs API calls that include SQL-, NoSQL, LDAP, OS and other **commands** that the API or backend behind it **blindly executes**.

HOW TO PREVENT

- 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.

OWASP TBILISI

API9:2019 - Improper Assets Management



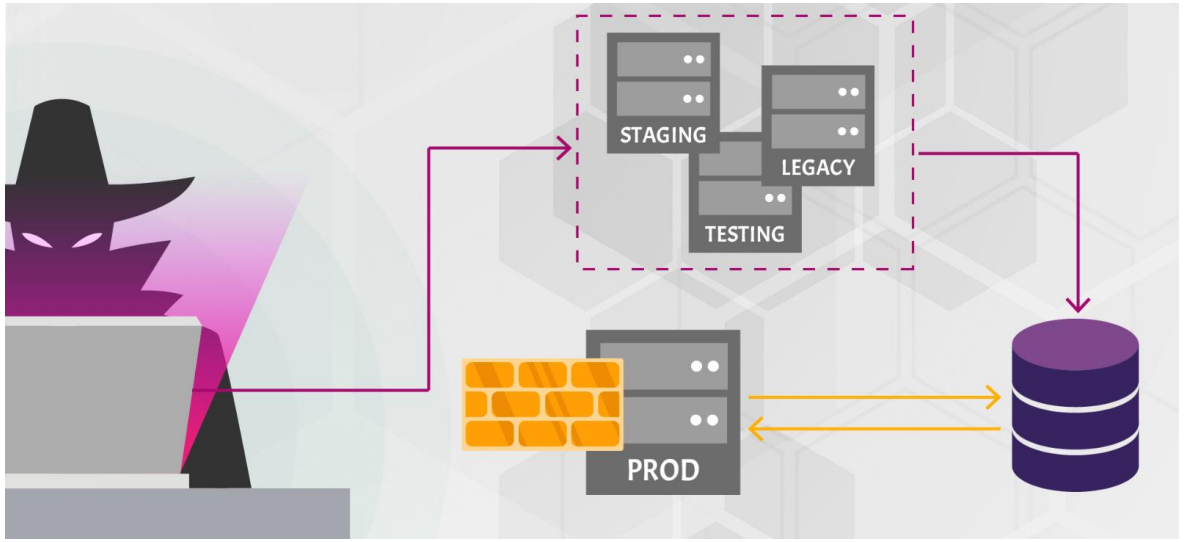
Attacker finds non-production versions of the API: such as **staging**, **testing**, **beta** or earlier versions - that are not as well protected, and uses those to launch the attack.

USE CASES

- DevOps, cloud, containers, K8S make having multiple deployments easy (Dev, Test, Branches, Staging, Old versions)
- Desire to maintain backward compatibility forces to leave old APIs running.
- Old or non-production versions are not properly maintained.
- These endpoints still have access to production data.
- Once authenticated with one endpoint, attacker may switch to the other.

OWASP TBILISI

Mitigation of API9:2019



Attacker finds non-production versions of the API: such as **staging**, **testing**, **beta** or earlier versions - that are not as well protected, and uses those to launch the attack.

HOW TO PREVENT

- Inventory all API hosts.
- Limit access to anything that should not be public.
- Limit access to production data. Segregate access to production and non-production data.
- Implement additional external controls such as API firewalls.
- Properly retire old versions or backport security fixes.
- Implement strict authentication, redirects, CORS, etc.

OWASP TBILISI

API10:2019 - Insufficient Logging & Monitoring

Lack of proper logging, monitoring, and alerting let attacks go unnoticed.

USE CASES

- Lack of logging, monitoring, alerting allow attackers to go unnoticed.
- Logs are not protected for integrity.
- Logs are not integrated into Security Information and Event Management (SIEM) systems.
- Logs and alerts are poorly designed.
- Companies rely on manual rather than automated systems.

OWASP TBILISI

Mitigation of API10:2019

Lack of proper logging, monitoring, and alerting let attacks go unnoticed.

HOW TO PREVENT

- 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.

OWASP TBILISI

OWASP API Security Top 10 2019

API1:2019 Broken Object Level Authorization
API2:2019 Broken User Authentication
API3:2019 Excessive Data Exposure
API4:2019 Lack of Resources & Rate Limiting
API5:2019 Broken Function Level Authorization
API6:2019 Mass Assignment
API7:2019 Security Misconfiguration
API8:2019 Injection
API9:2019 Improper Assets Management
API10:2019 Insufficient Logging & Monitoring

OWASP TBILISI

What's Next for :

- Developers
- Security Testers
- Organizations
- Application Managers

OWASP TBILISI

OWASP TBILISI

Questions ?

Thank you.

 /www-chapter-tbilisi/

 /OWASP-Tbilisi-Chapter

 /OWASP.Tbilisi