

HTTP Security Headers in Web APIs

Seminar Project

17 December 2019

Patrick Frischknecht

HTTP security headers

“HTTP response headers that your application can use to increase the security of your application. Once set, these HTTP response headers can restrict modern browsers from running into easily preventable vulnerabilities.”

OWASP Secure Headers Project

Support in browsers

	IE	Chrome	Safari	Firefox
Strict-Transport-Security	yes	yes	yes	yes
Content-Security-Policy	yes	yes	yes	yes
X-Frame-Options	yes	yes	yes	yes

Support in HTTP client libraries

	Glide	OkHttp	Volley	ION
Strict-Transport-Security	no	no	no	no
Content-Security-Policy	no	no	no	no
X-Frame-Options	no	no	no	no

Reasons for lacking support

Browsers

- must be able to access any site
- must provide HTTP (backwards compatibility)
- rich set of potentially harmful features (e.g., JavaScript execution)

API client libraries

- can restrict access to a set of whitelist pages
- can completely prevent HTTP connections
- offer limited features
- are not responsible for most potentially harmful features

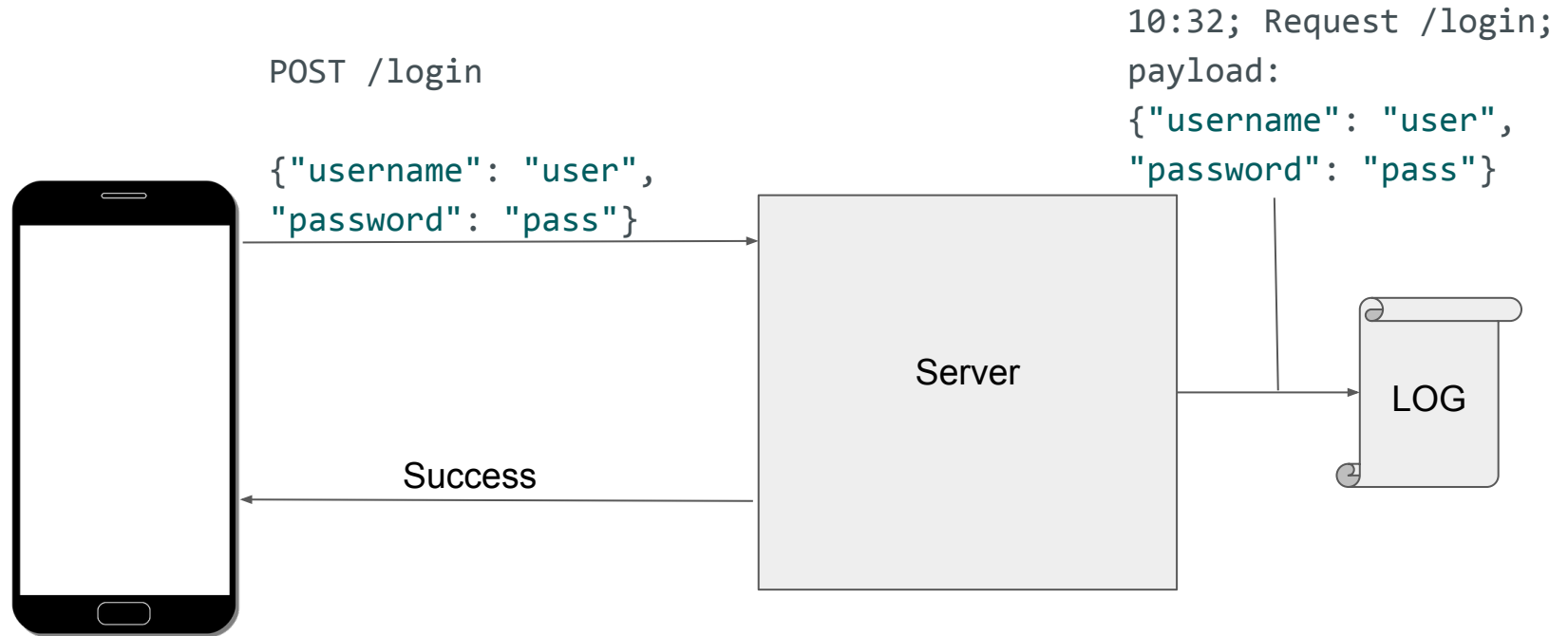
Is there more we can do?

What do we want to mitigate?

The most common threats:

- sensitive information leaks
- code injection attacks

Example 1: sensitive information leak



Persistence-Allowed header field

Syntax

Persistence-Allowed: any | only-hashed | none

Semantics

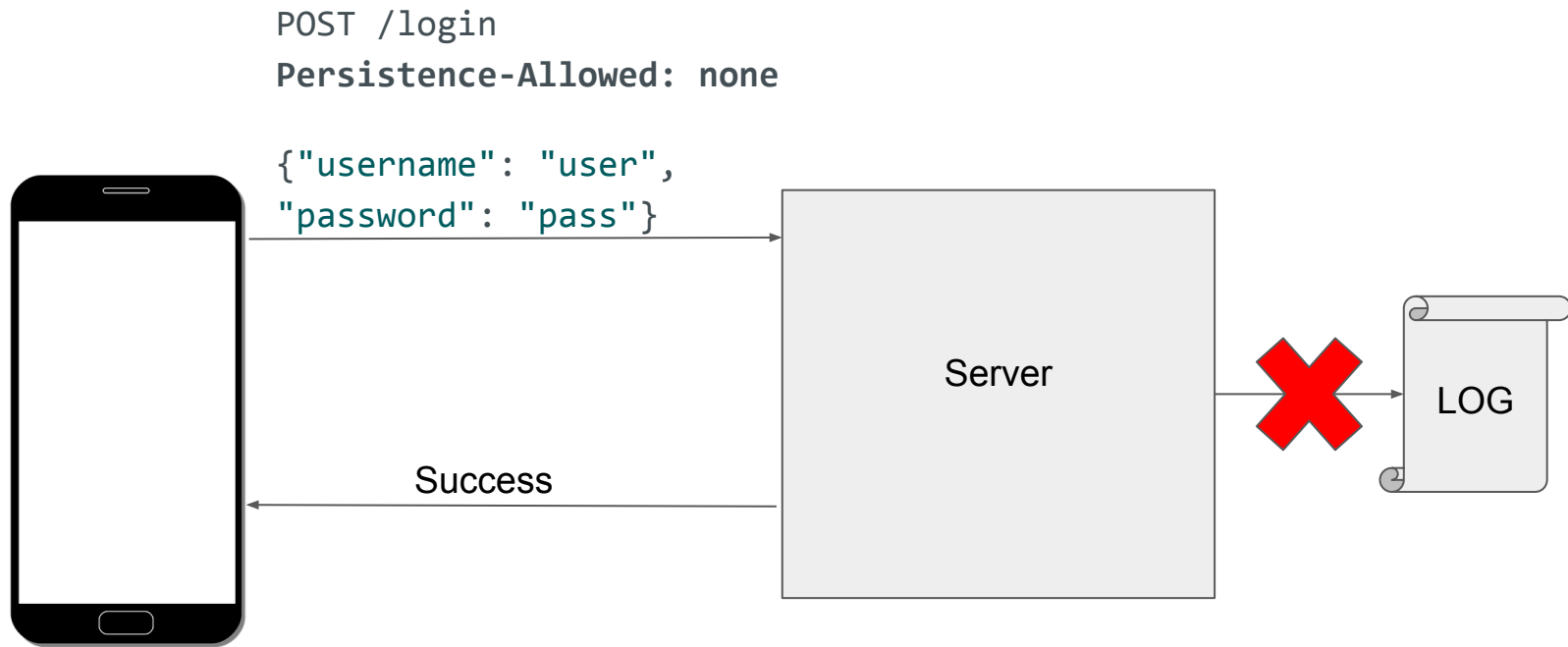
any allows to store content everywhere

only-hashed allows to store content only after hashing

none does not allow to store content
(content must reside in memory)

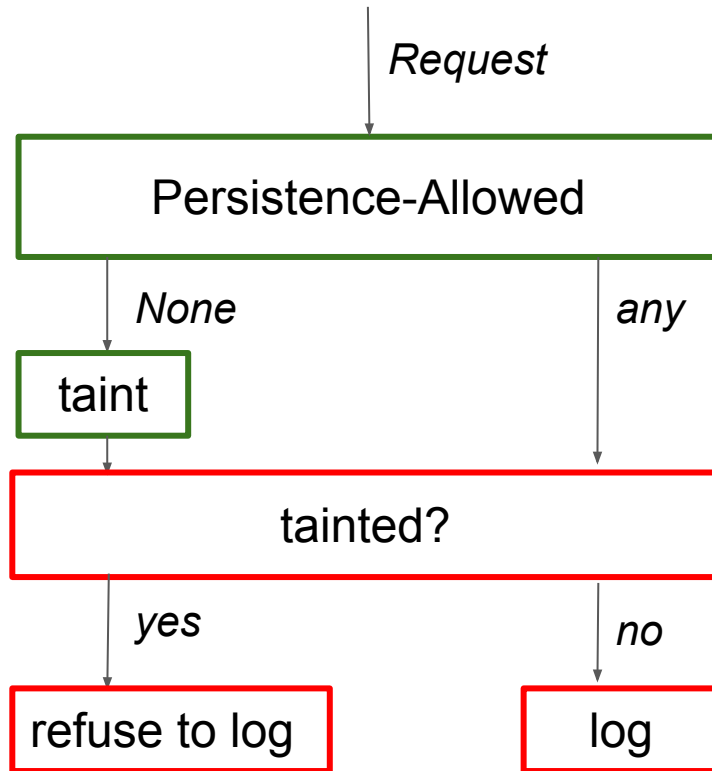
Example 1: sensitive information leak

using our header



Prototype demo

Prototype flow diagram



Persistence-Allowed middleware

logging middleware

Interpretation-Allowed header field

Syntax

Interpretation-Allowed: any | language1,language2 | none

Semantics

any	content can be interpreted/executed in any language
language1, . . .	content can be interpreted in the provided languages
none	content must not be interpreted

Example 2: Code injection

Request

apple

Server-side code

```
db.query("SELECT * FROM articles WHERE name='" + response.body + "'")
```

Result

An error is raised, since there is a potential SQL injection vulnerability

Example attack value: `10'; DROP TABLE 'articles`

Future work

- Implementation of framework or language runtime that supports both headers
- Evaluation on open-source and commercial projects

Conclusion

1. HTTP security headers are poorly supported by web API clients
2. Code injection attacks and data leaks are major threats
3. Proposed HTTP headers
 - a. `Persistence-Allowed` mitigates data leaks
 - b. `Interpretation-Allowed` mitigates code injection attacks