OWASP Privacy Toolkit

Martino Lessio

OWASP Italy @ Security Summit 2024

Milan - 21st March 2024

Agenda

- $ whoami
- The project
- Main features
- Next steps
- Q&A

Open: Everything at OWASP is radically transparent from our finances to our code.

Innovative: We encourage and support innovation and experiments for solutions to software security challenges.

Global: Anyone around the world is encouraged to participate in the OWASP community.

Integrity: Our community is respectful, supportive, truthful, and vendor neutral

$ whoami

Martino Lessio

Principal Software Security Consultant **@IMQ Minded Security**

- ex-Dev
- MAPT, WAPT, NPT, GOPT, *PT
- SCR
- Fixing Support

Off work: guido trattori e cambio pannolini.

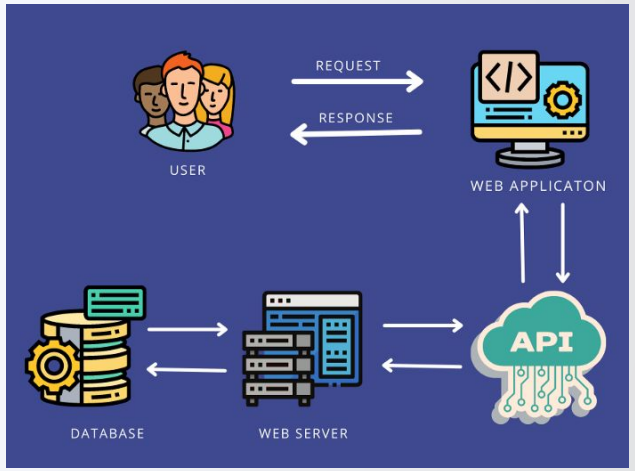
The Background

Developed by IMQ Minded Security within the TESTABLE project

- **H2020** EU-Funded Project
- Academic and Industry partners

The problem: privacy and data leaks

Modern web applications handle user's sensitive data.



But what is sensitive data in the context of a web app?

- Browsed websites (e.g. political or sexual contents)
- Websites categories (e.g. a specific campaign on change.org)
- URL/Path parameters
- Data manipulated in the DOM (e.g. user's personal data)


What is
this "Privacy" thing
that you
speak of?
bajio6401
march 2022


You don't believe in privacy,do you ?

A new OWASP project, again!

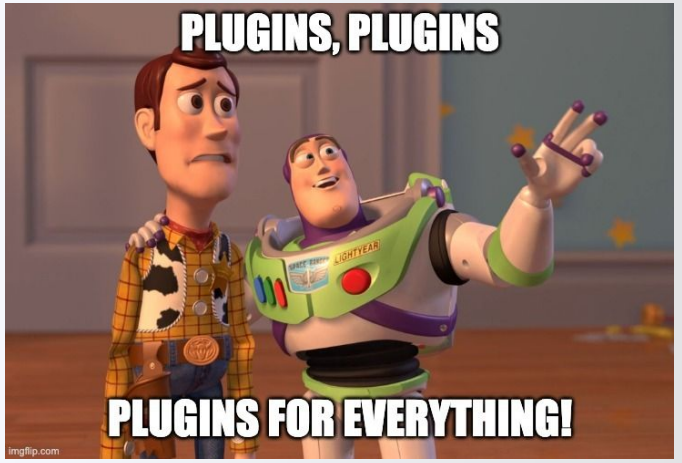




- *What is it*: a **browser extension**
- *Audience*: final **users**, **auditors** and **companies**
- *The goal*: bring **awareness, scoring,** and a report
- *Approach*: a modular project with an initial set of **plugins**

Authors: Stefano Di Paola, Martino Lessio *@IMQ Minded Security*

What we have? Plugins!



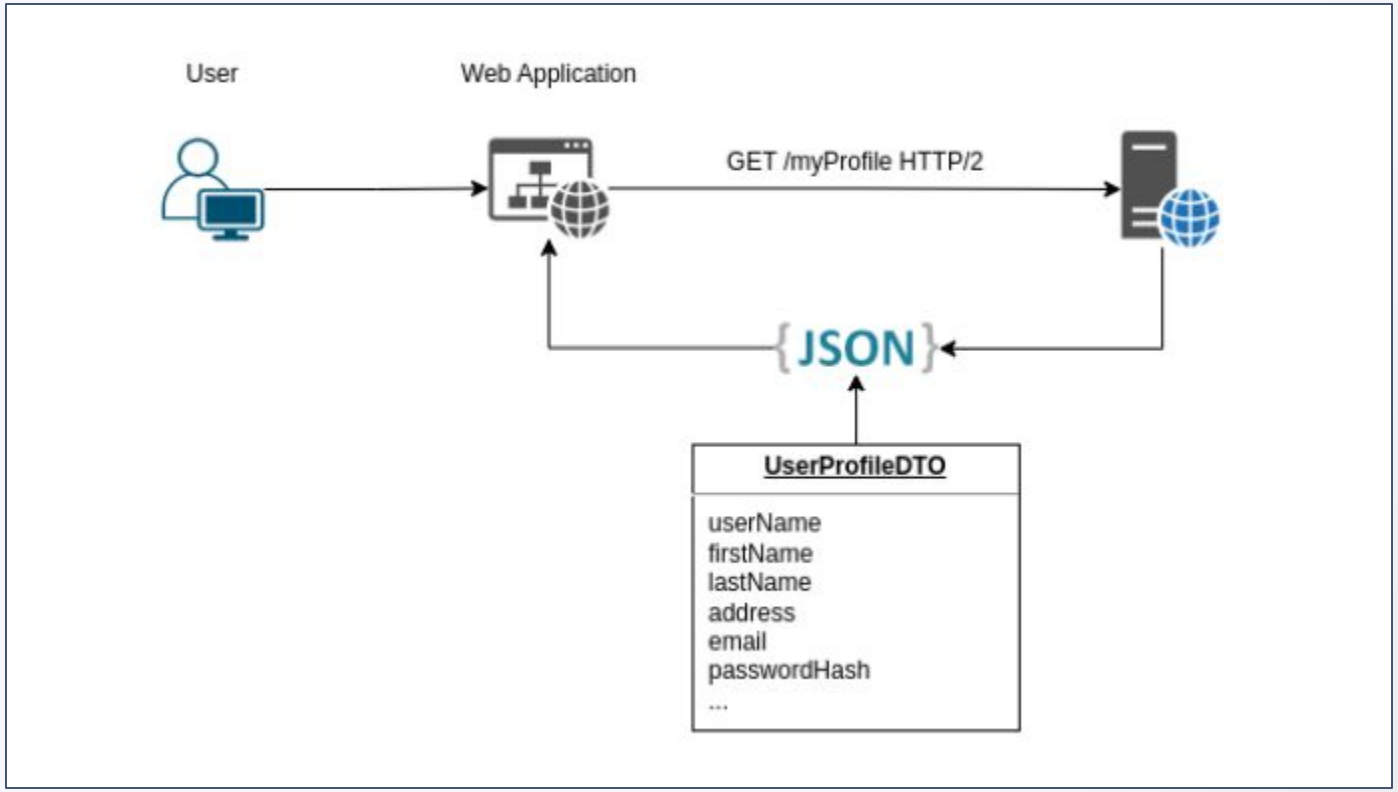
- Plugin - Data Oversharing — Status: ***Implemented***
- Referrer Leakage — Status: ***Implemented***
- Plugin - Globally Accessible Data — Status: ***Implemented***
- Third-party script positioning — Status: ***Implemented***
- Prototype Hijacking — Status: ***Implemented***
- Plugin - Abusable Data Storage — Status: ***Design***

Plugin - Data Oversharing

- **The problem:** detect all the overshared data b/w server and client.
- **OWASP Ref**: https://tinyurl.com/owasp-api3
- **Impact:** Web applications may receive/send too many information unnecessary to the app purposes.
- **Pattern Identification:**
  - Identify/Hook JSON deserialization APIs
  - Hook/Proxy Object data getters/setters
  - Correlate received data with used data (e.g. data tainting or I/O correlation)
  - Provide a score and a report on the data model with used and unused data

Implemented (Alpha)

Data Oversharing: The Concept



Oversharing: example

```
####################################
################RESULT##############
####################################
{
  'https://sampleapp.mindedsecurity.com/myProfile': {
    id: false,
    name: false,
    username: true,
    email: true,
    address: {
      street: false,
      suite: false,
      city: false,
      zipcode: false,
      geo: [Object]
    },
    phone: false,
    website: false,
    company: { name: false, catchPhrase: false, bs: false }
  }
}
```

Data Oversharing: the UI

< Back

Oversharing Plugin Report

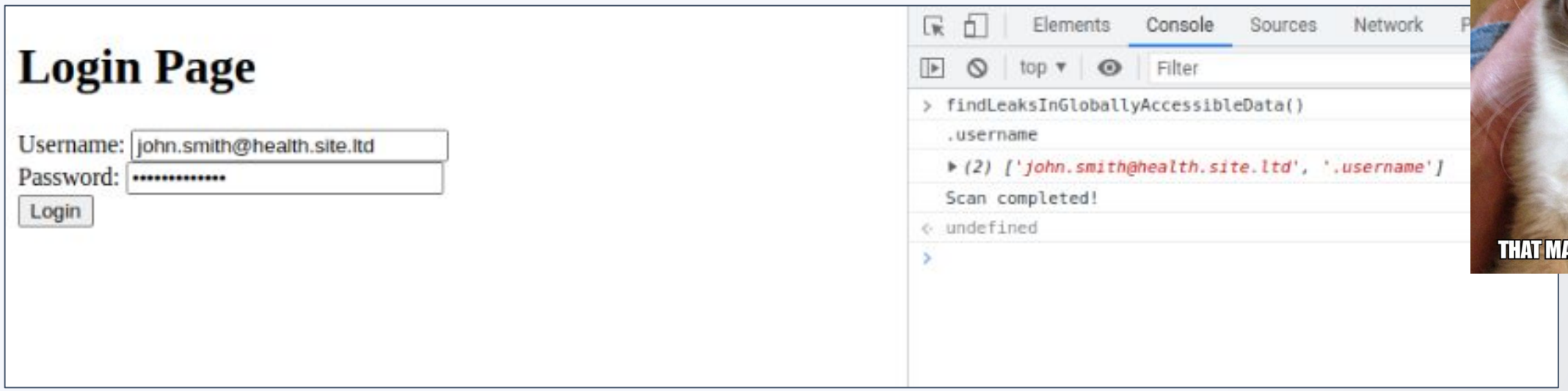
API Endpoint	API Data
	id× node_id× name× full_name× private× owner✓ html_url✓ description× fork× url× forks_url× keys_url× collaborators_url× teams_url× hooks_url× issue_events_url× events_url× assignees_url×

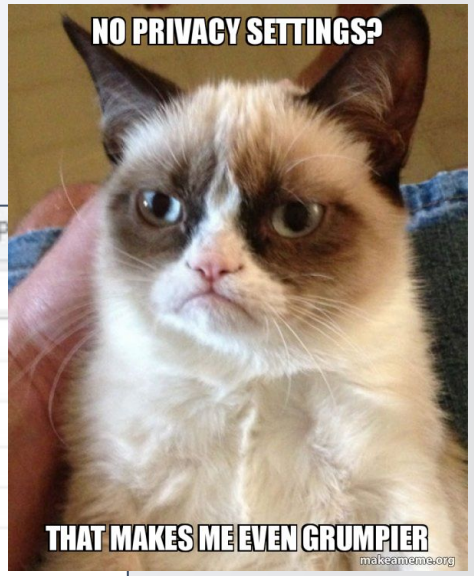
Plugin - Globally Accessible Data

- **The problem:** 3rd Party JS accessing sensitive information might affect its confidentiality.
- **OWASP Ref:** N/A
- **Impact**: The Application stores data in *globally accessible variables*.
- **Pattern Identification:**
  - List all the user defined variables into the DOM
  - Extract all the data
  - Infer/Ask sensitive data categorization by regexp etc (e.g. PII)
  - List all the sensitive data which can be accessed by arbitrary code

Implemented (Alpha)

Globally Accessible Data: example





Plugin - Referrer Leakage

- **The problem:** In cross-domain HTTP requests, the referrer header can leak information to third parties.
- **OWASP ref:** https://tinyurl.com/owasp-referrer
- **Impact**: A user is browsing a website interacting with external parties, which can get sensitive information about the user's habits.
- **Pattern Identification:**
  - Intercept all the outgoing HTTP requests
  - Check the Referer HTTP header
  - Rank the exposed information to each external actor

Implemented (Alpha)

Referrer Leakage: example

Plugin - Third-party Script Positioning

- **The problem:** If a third-party untrusted script is loaded and executed before the application's internal JS code, it may compromise the environment and data integrity.
- **OWASP ref:** https://tinyurl.com/owasp-3rdp
- **Impact**: a third-party script modifies DOM-level functions in order to steal sensitive data.
- **Pattern Identification:**
  - Identify all the loaded scripts source URLs
  - Check the top window URL against the script URL
  - Warn if an internal script is loaded after a third-party one

Implemented (Alpha)

Third-party Script Positioning: example

```
// Good
<script src="1st party JS"></script>
<script src="external 3rd party"></script>

// Bad
<script src="external 3rd party"></script>
<script src="1st party JS"></script>
```


```
▼WARNING! Best Practice says 3rd party scripts should be loaded after 1st party
scripts. video.repubblica.it https://video.repubblica.it/embed/metropolis/metropol
pm&type=sticky&w_t_pagehref=https://www.repubblica.it/#INLINE_SCRIPT
▼ {synch: Array(4)} i
  ▼ defer: Array(1)
    ▶ 0: {is_same_sld: false, requested_url: 'https://webcomponent.gedidigital.it/
      length: 1
    ▶ [[Prototype]]: Array(0)
  ▼ synch: Array(26)
    ▶ 0: {is_same_sld: true, requested_url: 'https://video.repubblica.it/embed/met
    ▶ 1: {is_same_sld: true, requested_url: 'https://video.repubblica.it/embed/met
    ▶ 2: {is_same_sld: false, requested_url: 'https://tlh.gedidigital.it/tlh.js?ma
    ▶ 3: {is_same_sld: true, requested_url: 'https://video.repubblica.it/embed/met
    ▶ 4: {is_same_sld: false, requested_url: 'https://webcomponent.gedidigital.it/
    ▶ 5: {is_same_sld: false, requested_url: 'https://scripts.kataweb.it/wt/wt.js?
    ▶ 6: {is_same_sld: false, requested_url: 'https://www.repstatic.it/cless/commo
    ▶ 7: {is_same_sld: false, requested_url: 'https://www.repstatic.it/cless/chann
    ▶ 8: {is_same_sld: false, requested_url: 'https://oasjs.kataweb.it/adsetup.rea
    ▶ 9: {is_same_sld: false, requested_url: 'https://cdn-gl.imrworldwide.com/conf
    ▶ 10: {is_same_sld: false, requested_url: 'https://cdn-gl.imrworldwide.com/nov
    ▶ 11: {is_same_sld: false, requested_url: 'https://static.chartbeat.com/js/cha
    ▶ 12: {is_same_sld: false, requested_url: 'https://www.googletagmanager.com/gt
    ▶ 13: {is_same_sld: false, requested_url: 'https://static.chartbeat.com/js/sub
    ▶ 14: {is_same_sld: false, requested_url: 'https://webcomponent.gedidigital.it
```

Plugin - Prototype Hijacking

- **The problem:** If a third-party untrusted script is loaded and the application code does not use a trusted reference to native functions, it may hijack the prototype of native functions and access sensitive data.
- **OWASP ref:** N/A
- **Impact**: a third-party script hooks or modifies DOM-level functions in order to steal sensitive data.
- **Pattern Identification:**
  - Identify all the direct invocations to a native method
  - Warn if the method access and invocation is on the same point

Implemented (Very very alpha, due to performance issues)

Prototype Hijacking: example

```
//Third Parts JS
var saved_join = Array.prototype.join;
Array.prototype.join = function (){
  //Collect Elements for "evil" purposes
  //..
  // then go on with the native method
  return saved_join.apply(this,arguments);
}

//1st part

(function (){

 var sensitive_data = ["XXXX","YYY"]; // Cannot be reached

 function treat_data(){
   // use sensitive_data
   return sensitive_data.join();
 }
})();
```

```
[Prototypes] GETTER WRAPPED
▶ {http://at.tack.er:3000/external.js?ssss22: {…},
EVIL INFO:  1
EVIL INFO:  iamasecret
 Malicious wrap22
[Prototypes] Join CALLED!!!!
▶ {http://at.tack.er:3000/external.js?ssss22: {…},
```

Plugin - Abusable Data Storage

- **The problem:** Unauthorized code such as 3rd Party JS accessing sensitive information might affect its confidentiality and integrity.
- **Scenario**: The Application stores data in globally accessible *storage*. (Eg localStorage, sessionStorage …
- **Pattern Identification:**
  - Hook the storage API with the Proxy pattern
  - Define patterns to identify potential sensitive data
  - Report information on where sensitive data is stored

Design

What we don't have? A(beautiful)UI

Next Steps

- User Friendly **UI** Design & Development
- Plugin Performance Optimization
- Identify **trusted data sources** and CDNs to minify FPs and background noise

New Plugins - Ideas

- Other Data Leakages:
  - Sensitive data in HTML DOM attributes
  - Sensitive data exfiltration through XHR request hooking

And also:

- Any plugin you are wondering now ->

CTA: Info, Testing, Joining, etc.

- Project will be **publicly available** in a short period (days)
- Anyone interested in joining the project is very welcome

SCAN ME

Q&A

Get in touch with us!

Email: info@mindedsecurity.com

Project page: https://owasp.org/www-project-privacy-toolkit/

Github repo *(coming soon)*: https://github.com/mindedsecurity

My contacts: scan the QR!

OWASP®

ITALY