Corso di Programmazione Web e Mobile

A.A. 2021-2022

SinK

Davide, Zorzella, 984617

Autore: Davide Zorzella
Ultima modifica: 12 Settembre 2022
Prima modifica: 23 Giugno 2022

SinK

Scova le tue vulnerabilità

1. INTRODUZIONE

Il progetto in questione si pone come scopo principale il trovare, analizzando la struttura del codice sorgente JavaScript, vulnerabilità note racchiuse sotto la grande famiglia denominata DOM XSS, Client XSS o, nella sua forma più estesa, DOM Cross Site Scripting.

SinK è un termine utilizzato in questa tipologia di attacchi ed identifica il punto esatto del Document Object Model (DOM) in cui possiamo osservarne gli effetti.

1.1 Breve Analisi dei requisiti

1.1.1 Destinatari

Il **pubblico** a cui il progetto è rivolto è un pubblico non ancora familiare con le tecnologie discusse e analizzate all'interno del software. Con un'interfaccia user-friendly e con un limitato numero di click del mouse è possibile infatti sfruttare un potente strumento di parsing e scanning che, una volta individuato in maniera del tutto autonoma ed automatica frammenti di codice Javascript nei siti richiesti, ricerca e riscontra all'interno pattern con vulnerabilità note.

La larghezza di **banda** richiesta non è eccessiva, dovendo scaricare semplici documenti ipertestuali. Data la natura frammentata di questo tipo di connessioni, non è generato un traffico di rete continuo ed invasivo.

La **tipologia di dispositivo** utilizzato è ininfluente per tutta la durata del processo, rispettando l'interfaccia i canoni standard del responsive design.

Il **linguaggio** utilizzato dal software è prettamente tecnico, salvo qualche eccezione doverosa, per permettere già da subito a chi ne sa poco (o addirittura è la prima volta che si approccia a questo mondo) di fare pratica con termini come Server, IP, HTML, Javascript, Parsing, Pinging etc.

Le **motivazioni** che hanno portato alla creazione di questo software risiedono nella necessità intrinseca di ogni programmatore o gestore di servizi di ottenere report di sicurezza e affidabilità delle piattaforme web nel più breve tempo possibile, in modo da prevenire potenziali attacchi e mitigare vulnerabilità già sfruttate in passato.

Secondo il modello **Marcia Bates**, che colloca sull'asse verticale il grado di **consapevolezza** circa l'oggetto ricercato (quanto siamo in grado di specificare a parole il nostro bisogno) mentre sull'asse orizzontale il grado di **volontarietà** della ricerca (attiva o passiva), possiamo collocare SinK nella

categoria **Monitoring**, dato che sappiamo cosa stiamo cercando ma, essendo il processo automatizzato e automatico come già precedentemente specificato, subiamo una ricerca passiva.

1.1.2 Modello di valore

Il prodotto in questione rientra pienamente nella categoria **SaaS** (Software as a Service), ovvero *'un modello di servizio del software applicativo realizzato da un produttore che mette a disposizione un programma, direttamente o tramite terze parti, con modalità telematiche come ad esempio un'applicazione web.'* *Wikipedia*

SinK non è solo un tool per scovare vulnerabilità note a precedenti, ma permette tutta una serie di **operazioni di supporto** all'analisi e alla stesura di opportuni report, come il salvataggio su database delle vulnerabilità già scoperte, il download di risorse utili ma esterne alla pagina, un dettagliato recap del report appena generato che contiene informazioni utili sull'IP del server ospitante le risorse e il codice HTML in chiaro. E i risultati si vedono subito.

Il **valore economico** del prodotto non è stimabile direttamente, ma cresce in rapporto al maggior utilizzo che se ne fa. Attualmente lo si vuole improntare verso una direzione **Freemium**, ovvero con una versione completamente gratuita, ma che offre funzionalità di base limitate rispetto ad una più soddisfacente e completa versione a pagamento.

1.1.3 Flusso dei dati

Ottenere i dati

Una volta entrati nella piattaforma, per iniziare a scansionare in cerca di vulnerabilità DOM XSS, bisognerà per prima cosa indicare quali sono gli obiettivi target dell'analisi. Per far fronte a ciò bisognerà inserire nell'apposita casella di input uno o più link a risorse connesse al World Wide Web, rintracciabili dunque tramite url.

Il sistema, una volta scelti i bersagli, autonomamente scoverà quali tra queste risorse è online, collegata in rete e dunque raggiungibile, e quali invece non sono disponibili al momento, il tutto tramite richieste **XHttp**, gestite da **Web Workers**.

Possiamo ora scegliere singolarmente quali scansionare, cliccandone il relativo bottone, il quale ci porterà su una nuova pagina.

Una volta atterrati, verrà stilato un resoconto del report, frutto dell'elaborazione lato server del codice Javascript. Il server infatti, durante il caricamento della pagina, si è occupato di raccogliere le risorse linkate all'interno dell'HTML del nostro obiettivo, di eseguire un parsing del codice per ottenere un **Abstract Syntax Tree** (**AST**) e di ricercare all'interno di questa nuova struttura dati pattern presenti in un vocabolario, che poi inoltrerà al client. Tutte queste chiamate **asincrone** vengono gestite lato client da Web Workers.

Nel report è presente un resoconto sulle informazioni pubbliche ricavabili dall'indirizzo IP, grazie al servizio API gratuito https://ip-api.com/.

Archiviare i dati

L'archiviazione avviene in remoto, tramite il conosciuto e ben documentato **DataBase Management System MongoDB**. Il DBMS in questione archivia i dati in formato **JSON**, un semplice formato per lo scambio di dati, indipendente dal linguaggio di programmazione che lo utilizza, facile da leggere e scrivere e leggero.

Lato client invece gli unici dati salvati, mediante localStorage messo a disposizione da 'window' del browser, sono gli indirizzi url degli ultimi 3 siti scansionati che abbiano riportato almeno 1 vulnerabilità. L'add-on non ha scopo pratico, serve solo come piccolo recap delle ultime sessioni di lavoro.

Pubblicare i dati

I dati salvati su MongoDB vengono resi pubblici gratuitamente tramite **GUI** (interfaccia web) alla pagina 'reports' o tramite **API**. Quest'ultima offre due principali modalità di funzionamento: può infatti restituire gli ultimi 10 report tutti assieme o, tramite un id specificato nella richiesta, restituire soltanto quello ad esso associato.

Una volta archiviati i dati non è possibile però aggiornarli modificandoli, dato che la piattaforma non è pensata per offrire questo tipo di interazione. Per ottenere una nuova versione più recente bisognerà generare un nuovo report.

1.1.4 Aspetti Tecnologici

Le tecnologie utilizzate sono qui sotto riportate senza un ordine preciso:

(1) NodeJS

Framework Javascript per la scrittura di codice lato server

(2) Javascript

Linguaggio di scripting lato client utilizzato per rendere interattive le pagine web

(3) CSS / SCSS

Cascading Style Sheets, linguaggio utilizzato per definire la formattazione degli elementi di una pagina web HTML

Superset CSS: contiene tutte le funzionalità di CSS ma allo stesso tempo ampliato per includere anche le funzionalità di Sass

(4) HTML / EJS

HyperText Markup Language, linguaggio a marcatori per ipertesti, fondamento per la scrittura della struttura di un documento web

Embedded Javascript, template engine per express.js

(5) MongoDB

DBMS noSql opensource. In quanto tale utilizza un modello di dati non relazionali e orientato ai documenti e un linguaggio di query non strutturato

(6) BootStrap

Una raccolta di strumenti liberi per la creazione di siti e applicazioni web

(7) API

Application Programming Interface, ovvero un insieme di definizioni e protocolli per la creazione e l’integrazione di software applicativi

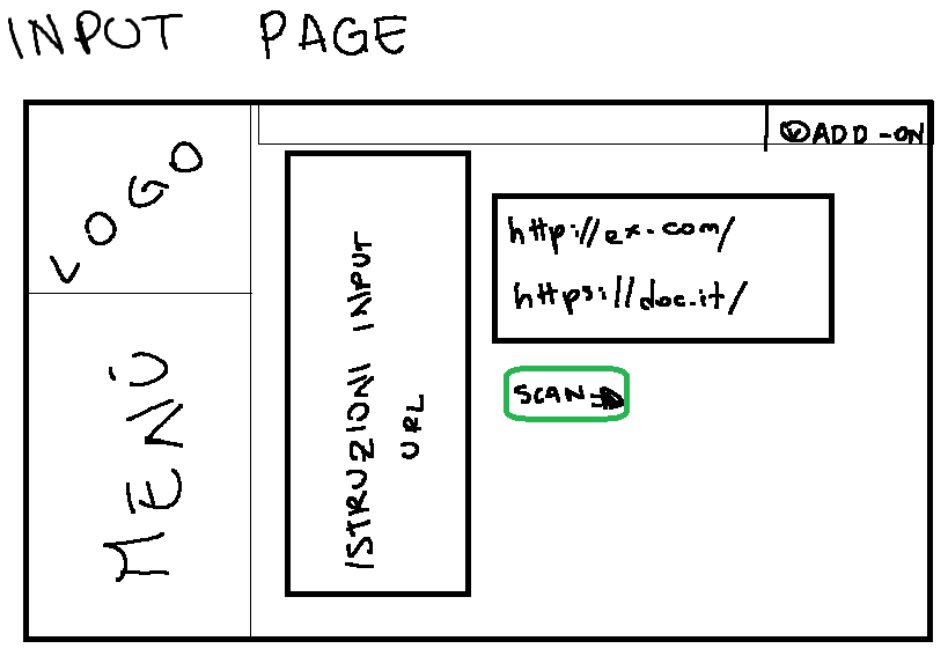
(8) JSON

Formato di serializzazione basato su testo per lo scambio di dati, principalmente tra server e applicazione web

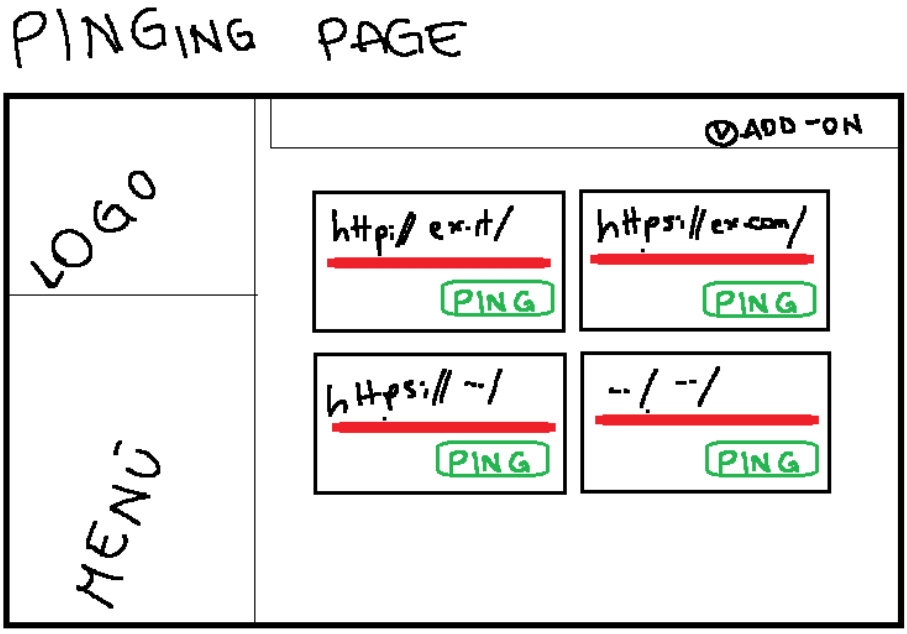
Autore: Davide Zorzella
Ultima modifica: 12 Settembre 2022
Prima modifica: 23 Giugno 2022

2. INTERFACCE

(1) Pagina di input delle risorse web da scansionare

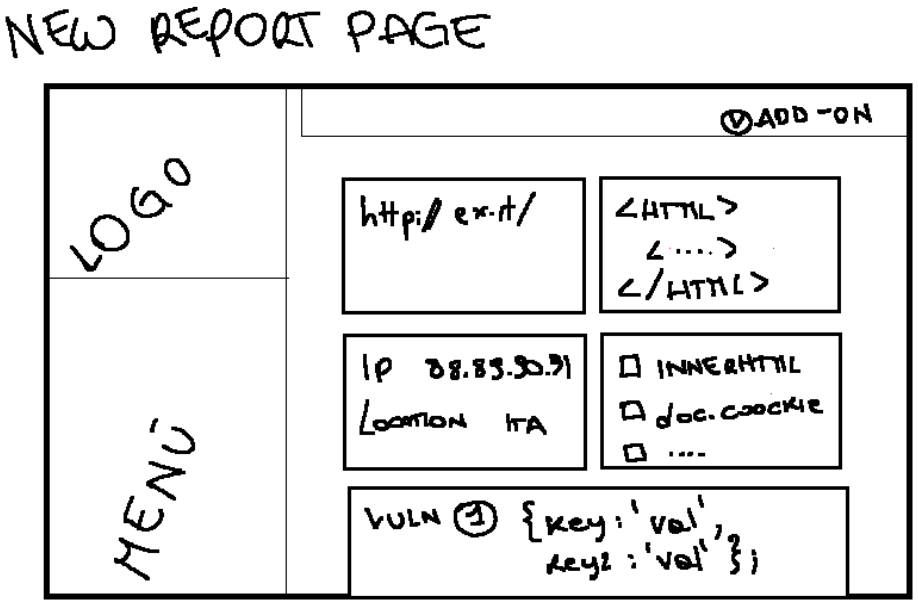


(2) Pagina di pinging dei vari server

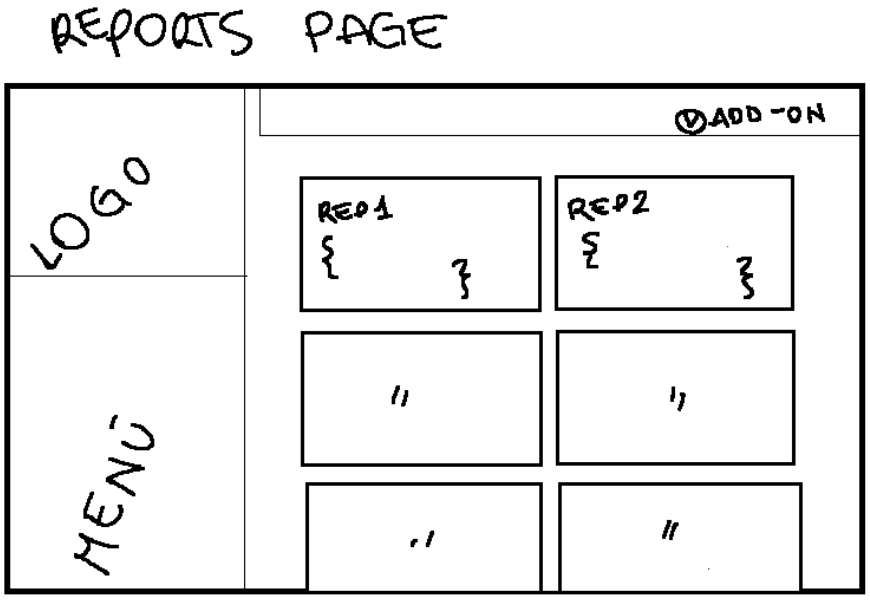


Autore: Davide Zorzella
Ultima modifica: 12 Settembre 2022
Prima modifica: 23 Giugno 2022

(3) Pagina di output del singolo report



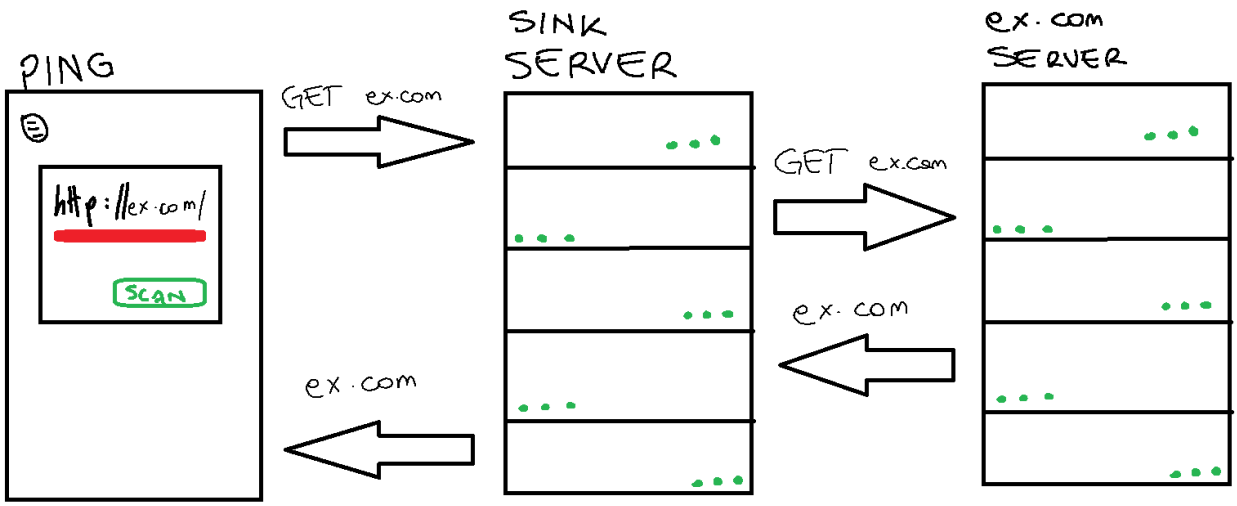
(4) Pagina di output degli ultimi 10 report



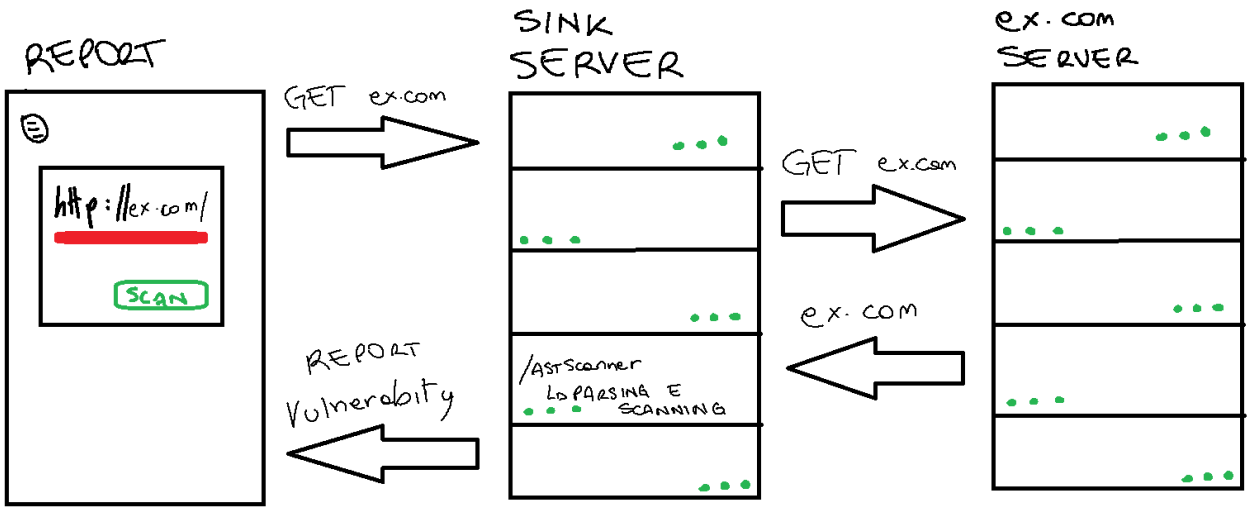
Autore: Davide Zorzella
Ultima modifica: 12 Settembre 2022
Prima modifica: 23 Giugno 2022

3. ARCHITETTURA

Ping

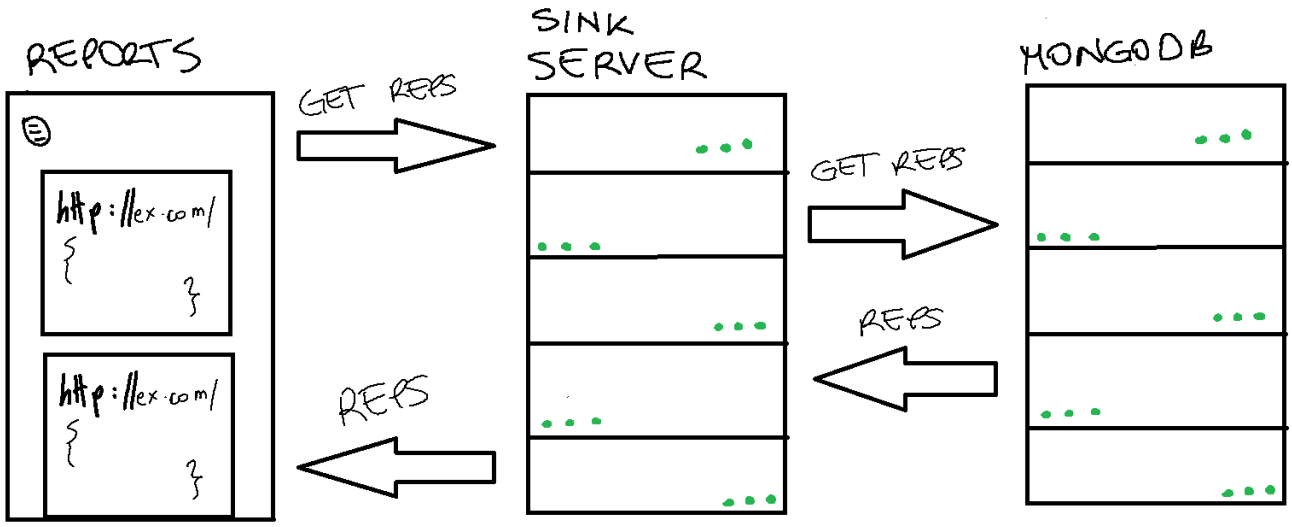


Report



Autore: Davide Zorzella
Ultima modifica: 12 Settembre 2022
Prima modifica: 23 Giugno 2022

Reports



Autore: Davide Zorzella
Ultima modifica: 12 Settembre 2022
Prima modifica: 23 Giugno 2022

4. CODICE

4.1 API

Collegamento ai servizi erogati da IP-API.com per ottenere informazioni utili a conoscere la posizione del server e altri dati riconducibili al suo IP.

```
let url = new URL(document.getElementById('url').value);
fetch('http://ip-api.com/json/' + url.host)
  .then((response) => response.json())
  .then((data) => {
    document.getElementById('apiQuery').textContent = data.query
    if (data.status == 'fail') return
    document.getElementById('apiStatus').textContent = data.status
    document.getElementById('apiCountry').textContent = data.country
    document.getElementById('apiCountryCode').textContent = '(' + data.countryCode + ')'
    document.getElementById('apiRegion').textContent = data.region
    document.getElementById('apiRegionName').textContent = data.regionName
    document.getElementById('apiCity').textContent = data.city
    document.getElementById('apiZip').textContent = data.zip
    document.getElementById('apiLat').textContent = data.lat
    document.getElementById('apiLon').textContent = data.lon
    document.getElementById('apiTimezone').textContent = data.timezone
    document.getElementById('apiIsp').textContent = data.isp
    document.getElementById('apiOrg').textContent = data.org
    document.getElementById('apiAs').textContent = data.as
  })
```

Autore: Davide Zorzella
Ultima modifica: 12 Settembre 2022
Prima modifica: 23 Giugno 2022

4.2 NODEJS

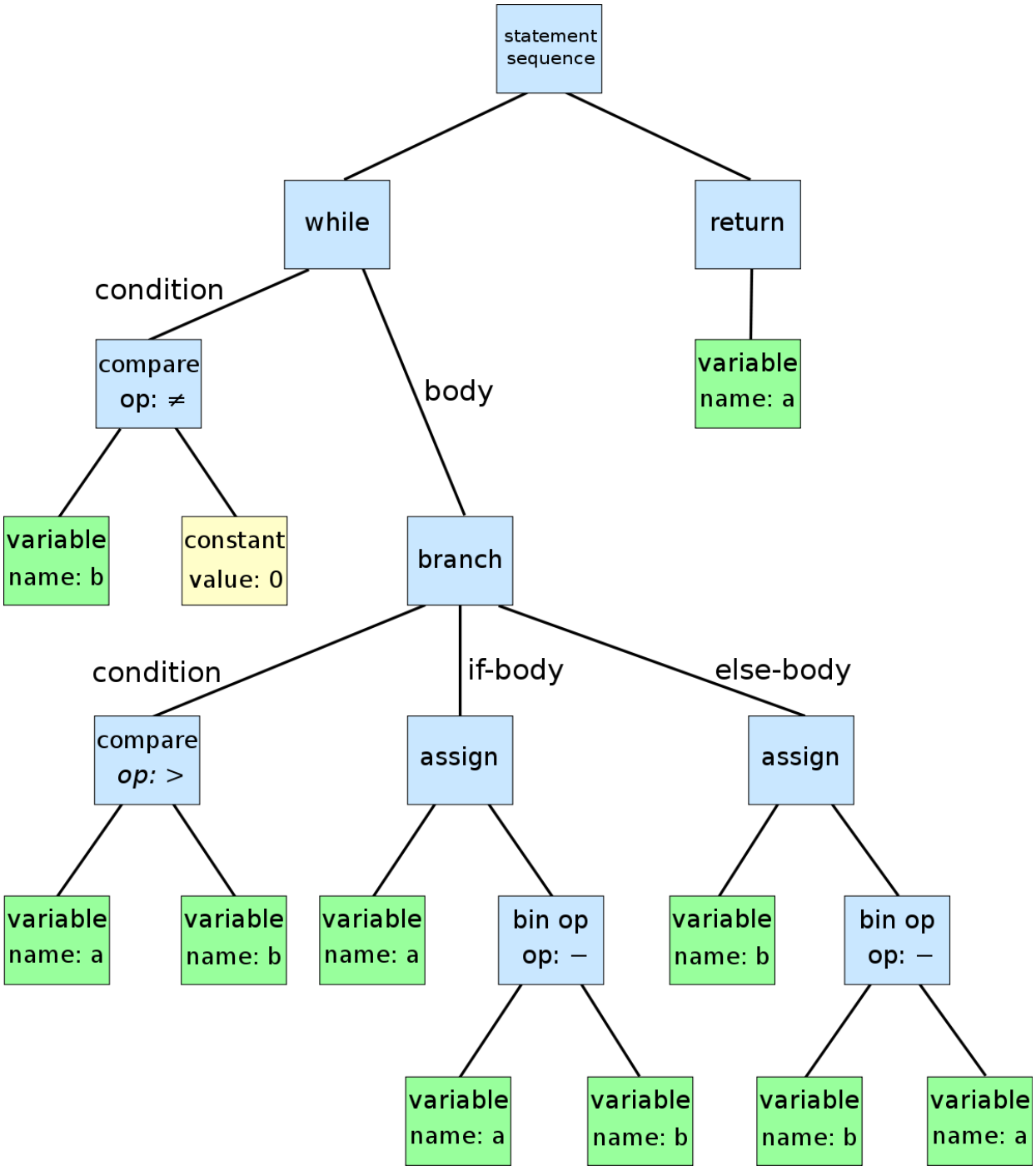
API che dato codice Javascript tramite POST request ritorni le vulnerabilità trovate all'interno

```
router.post('/ASTScanner', async (req, res) => {
  var sinks = [
    'write',
    'writeln',
    'insertAdjacentHTML',
    'innerHTML'
  ], vulns = [];
  try {
    const AST = parse(req.body.data);
    traverse(AST, {
      enter(path) {
        sinks.forEach(element => {
          if (path.isIdentifier({ name: element })) {
            // Ho trovato un sinks, non importa che sia davvero manipolabile a mio piacere
            var obj = {
              url: req.body.targetUrl,
              Node: {
                type: 'Identifier',
                loc: {
                  start: {
                    line: path.node.loc.start.line,
                    column: path.node.loc.start.column,
                    index: path.node.loc.start.index
                  },
                  end: {
                    line: path.node.loc.end.line,
                    column: path.node.loc.end.column,
                    index: path.node.loc.end.index
                  },
                },
                name: path.node.name
              }
            }
            // Create schema vuln
            var vuln = new Schema(obj);
            // Save it to mongodb
            vuln.save();
            // push into the array
            vulns.push(vuln);
          }
        });
      }
    });

    res.status(200).send(vulns);
  } catch (err) {
    res.status(500).send(err);
  }
})
```

Autore: Davide Zorzella
Ultima modifica: 12 Settembre 2022
Prima modifica: 23 Giugno 2022

Esempio di AST:



Autore: Davide Zorzella
Ultima modifica: 12 Settembre 2022
Prima modifica: 23 Giugno 2022

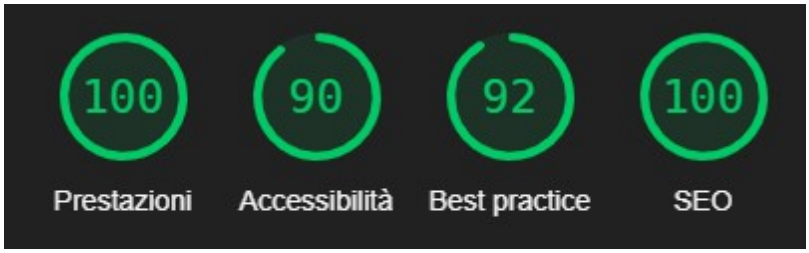
5. PRESTAZIONI E LIGHTHOUSE

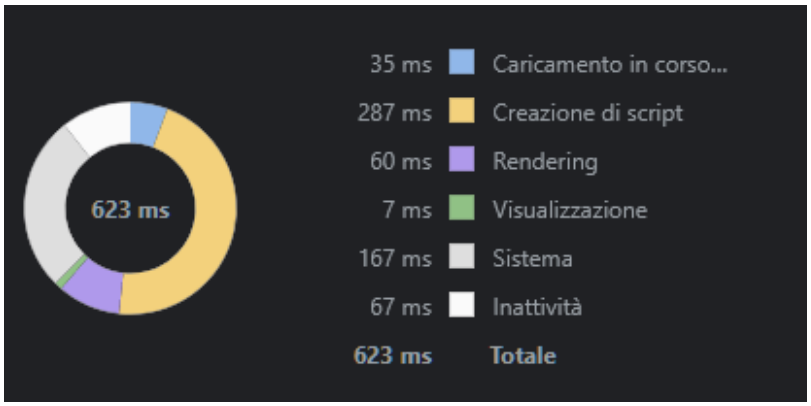
Viene presa come riferimento la risorsa web 'https://www.example.com/' disattivando l'uso dei dati salvati in cache

Le prestazioni vengono calcolate utilizzando lo strumento per sviluppatori 'Prestazioni' messo a disposizione da google chrome Versione 105.0.5195.127 (Build ufficiale) (a 64 bit)

Tutti i report dettagliati sono allegati nella cartella della documentazione

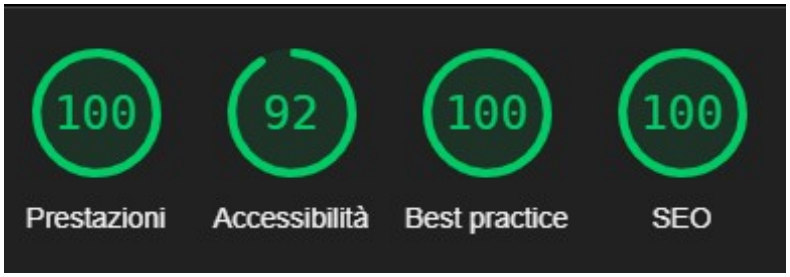
(1) Pagina di input delle risorse web da scansionare

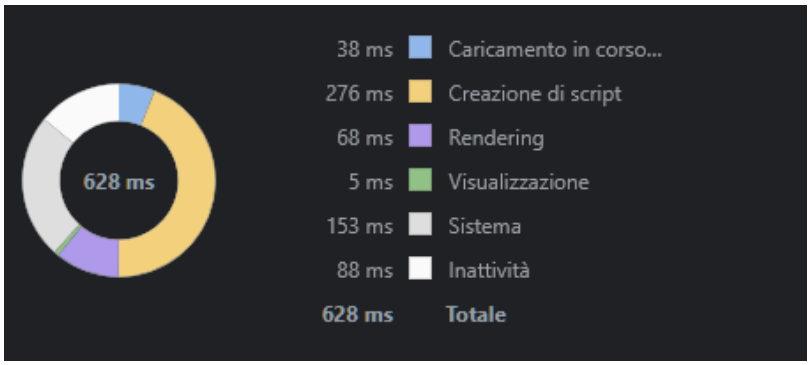




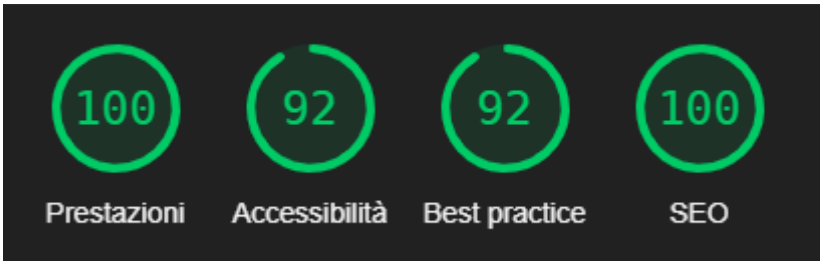
Autore: Davide Zorzella
Ultima modifica: 12 Settembre 2022
Prima modifica: 23 Giugno 2022

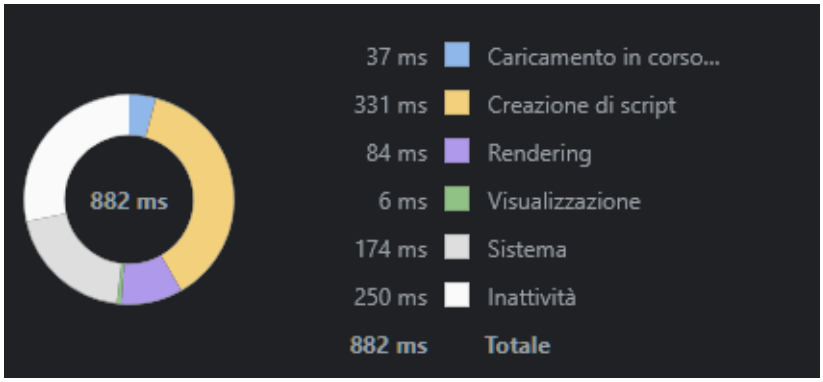
(2) Pagina di pinging dei vari server





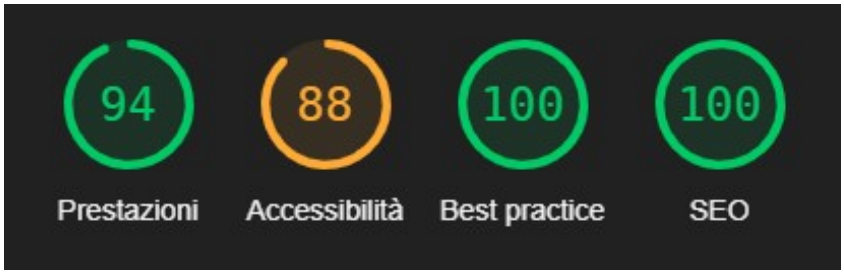
(3) Pagina di output del singolo report

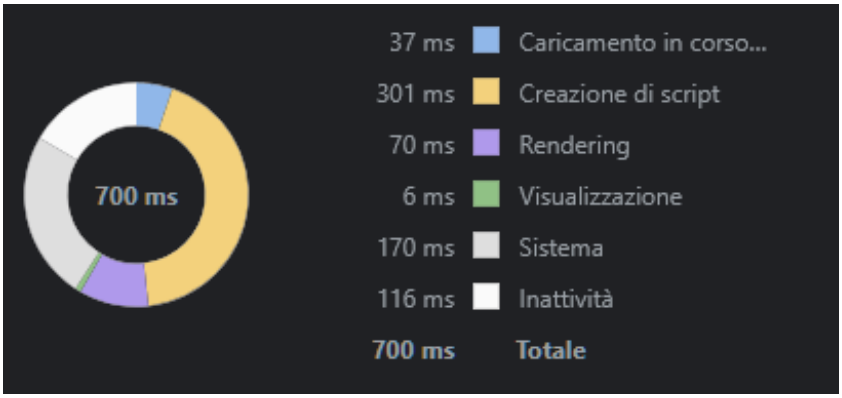




Autore: Davide Zorzella
Ultima modifica: 12 Settembre 2022
Prima modifica: 23 Giugno 2022

(4) Pagina di output degli ultimi 10 report





Autore: Davide Zorzella
Ultima modifica: 12 Settembre 2022
Prima modifica: 23 Giugno 2022

6. NOTA BIBLIOGRAFICA E SITOGRAFICA

(1) NodeJS Documentation, https://nodejs.org/it/docs/

(2) StackOverflow, https://stackoverflow.com/

(3) MongoDB Documentation, https://www.mongodb.com/docs/

(4) GitHub, https://github.com/

(5) IP-API, https://ip-api.com/

(6) DarkPan Template, https://htmlcodex.com/bootstrap-5-admin-template

(7) Babel/parser, https://babeljs.io/docs/en/babel-parser

(8) Babel/traverse, https://babeljs.io/docs/en/babel-traverse

Autore: Davide Zorzella
Ultima modifica: 12 Settembre 2022
Prima modifica: 23 Giugno 2022