The Remote on the Local

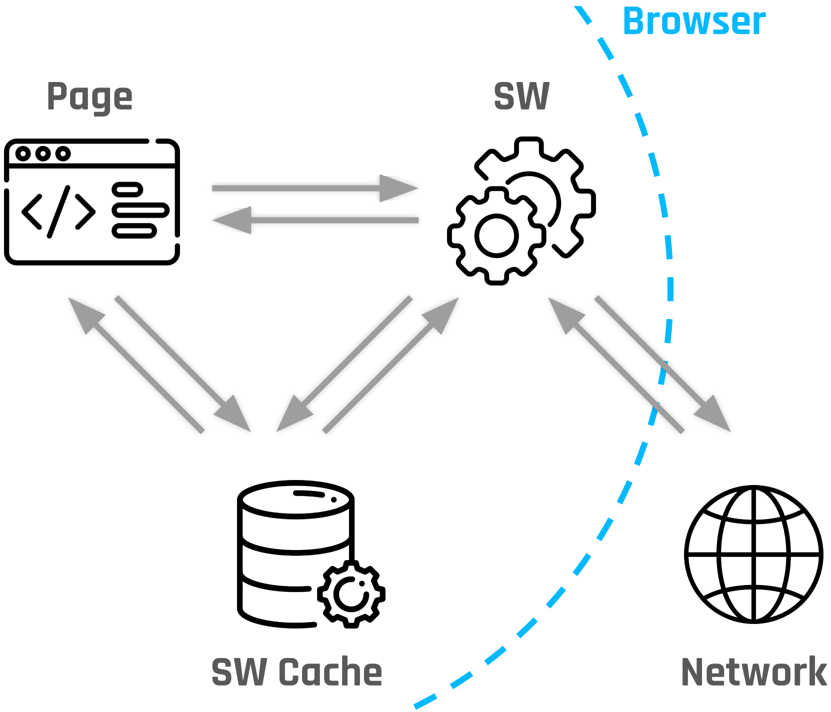
Exacerbating Web Attacks Via Service Workers Caches

Marco Squarcina (TU Wien)
🐦 @blueminimal

15th IEEE Workshop on Offensive Technologies. May 27, 2021

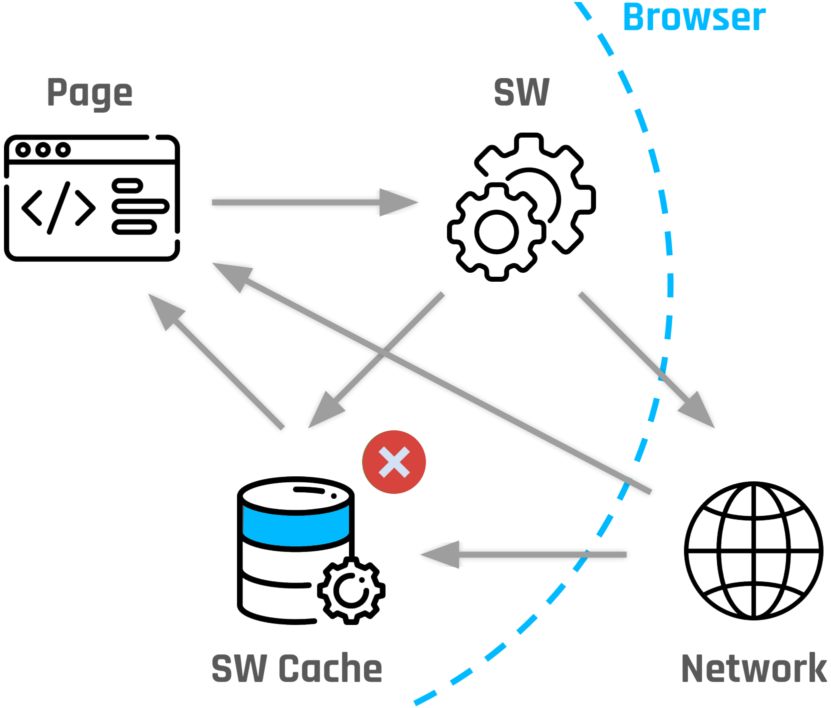
Joint work with
Stefano Calzavara (Università Ca' Foscari Venezia & OWASP)
Matteo Maffei (TU Wien)

Service Workers



- Key enabler of **PWAs**
- Client-side **web application proxies** able to intercept HTTP requests
- **Cache API** allows to store HTTP responses, **offline capabilities**
- SW execute in a **separate context**, no direct DOM access
- Operate based on **origin** and **path,** event-based activation

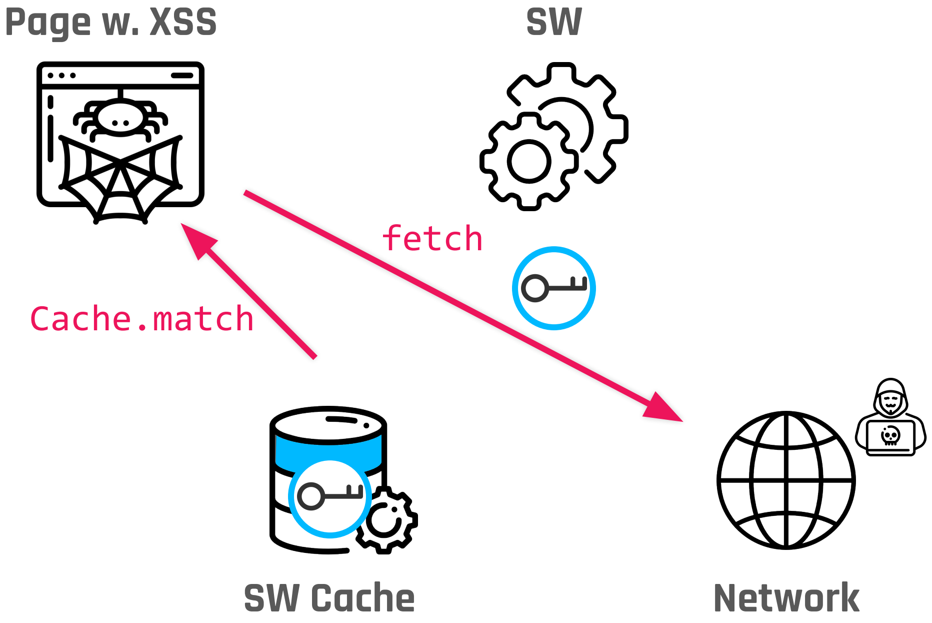
Cache-First/Offline-first Pattern



```
1 self.addEventListener('fetch', (e) => {
2   e.respondWith(
      caches.match(e.request).then((r) => {
3       return r ||
4       fetch(e.request).then((res) => {
5         return caches.open('static').then(
            (cache) => {
              cache.put(e.request,
                res.clone());
6             return res;
            }
          );
        });
      })
    );
  });
```

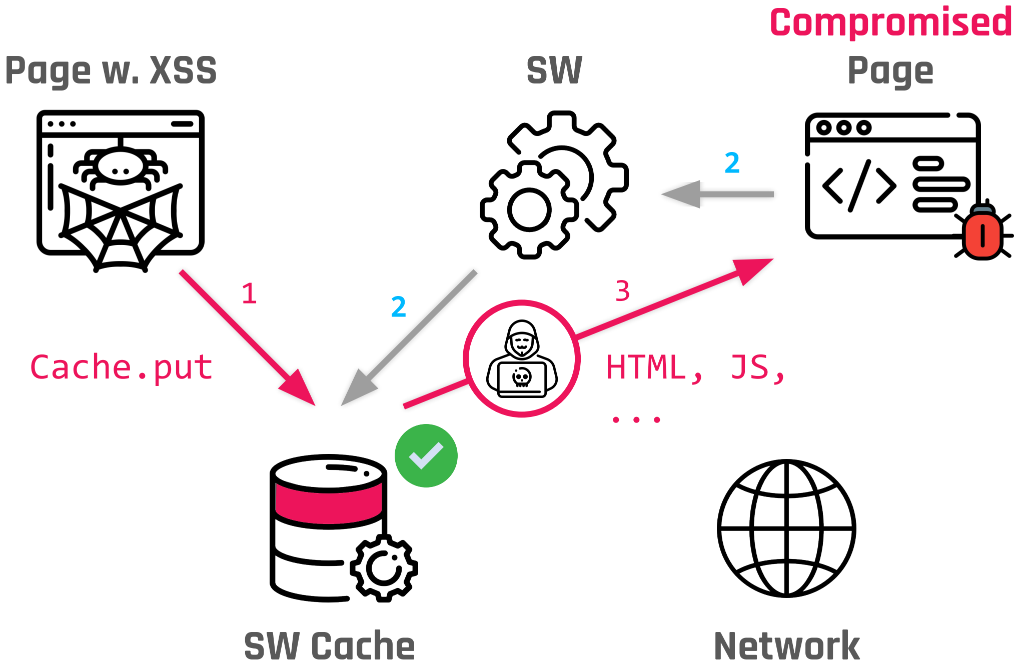
SW Code

Secret Exfiltration



- SW **Cache can be accessed** also from **scripts running in the page**
- Web attacker with XSS on a page can **leak cached secrets** bound to the **entire origin**!
- This includes secrets left over from a **previous session** like
  - personally identifiable information
  - passwords
  - security tokens
  - multimedia content

Content Corruption



- Cache entries can also be arbitrarily **modified** and **forged**
- An attacker can modify a response to
  - **Inject malicious JS** (e.g. keylogger) (by editing a cached JS file or by injecting a script in a page)
  - **Tamper HTTP response headers**
- Similar to **persistent client-side XSS**
  - Reflected XSS → **persistent** attack
  - Denial of Service (**DoS**)
  - **Amplification** of the attack surface

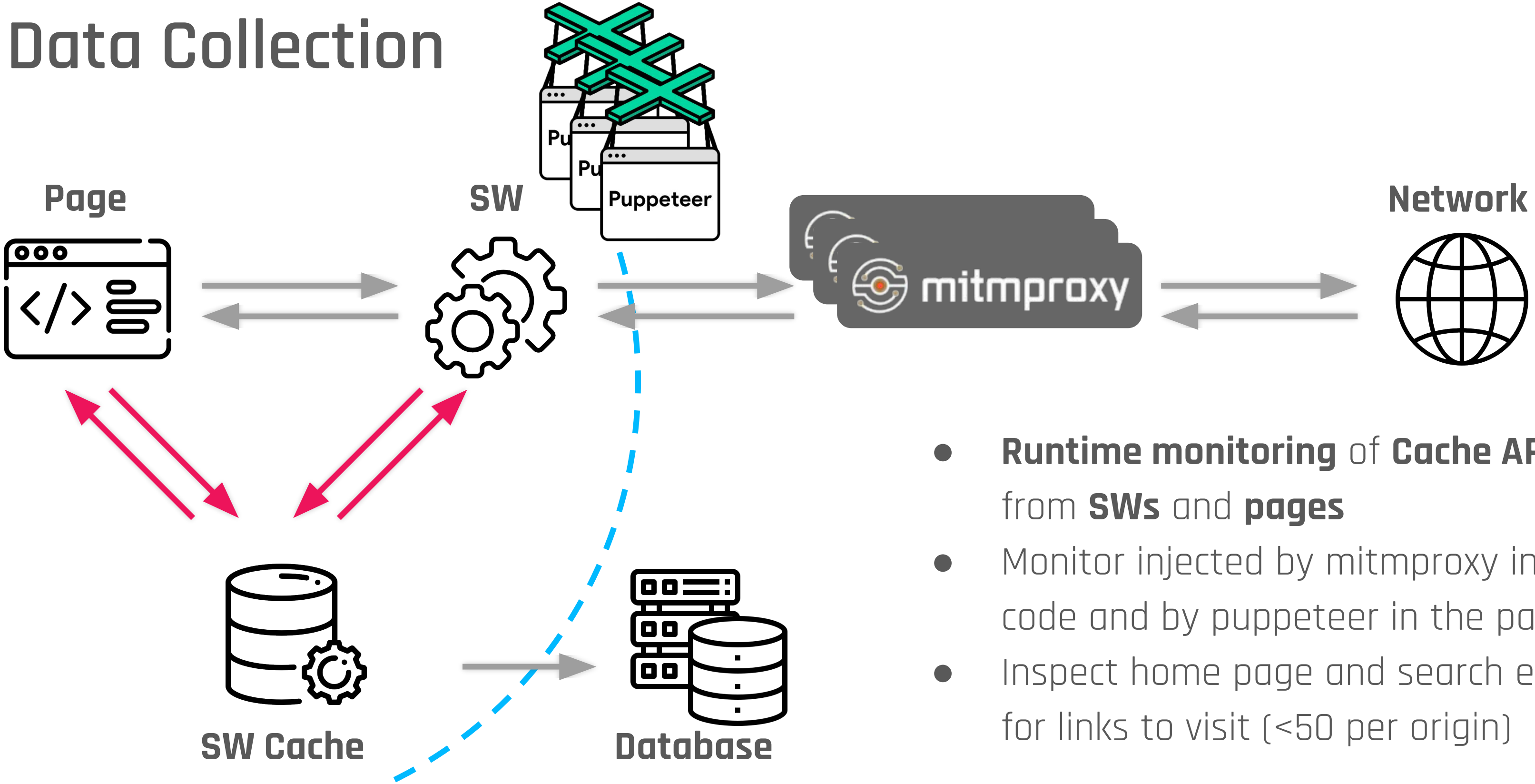
PITM on HTTP responses

- **Inspect and modify response** objects, including **HTTP headers**
- Not possible with a traditional XSS, more similar to HTTP **response splitting attack**

- **Framing**
  - Disable CSP `frame-ancestors` and `X-Frame-Options`
- **Privilege Escalation**
  - Disable ~~Feature~~ Permission Policy to access webcam, microphone, geolocation, etc.

- **Break Isolation**
  - Avoid SOP enforcement by removing CSP `sandbox` directive and iframe attribute
- **Bypass Defensive Programming**
  - Void the robustness of JS code (Constants, Frozen Objects, Sealed Objects, …)

Data Collection



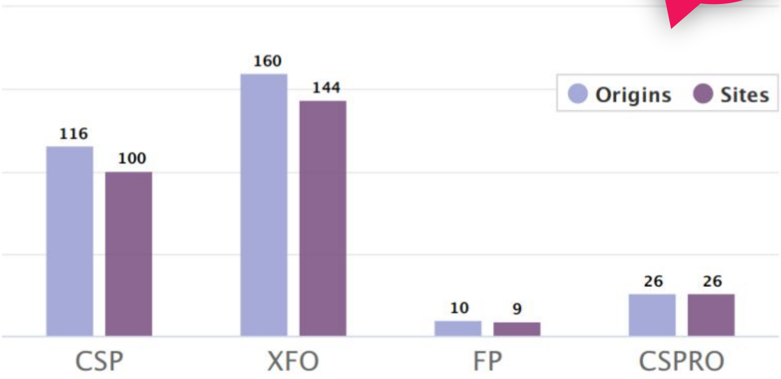
- **Runtime monitoring** of **Cache API calls** from **SWs** and **pages**
- Monitor injected by mitmproxy in SW code and by puppeteer in the pages
- Inspect home page and search engines for links to visit (<50 per origin)

Large Scale Assessment

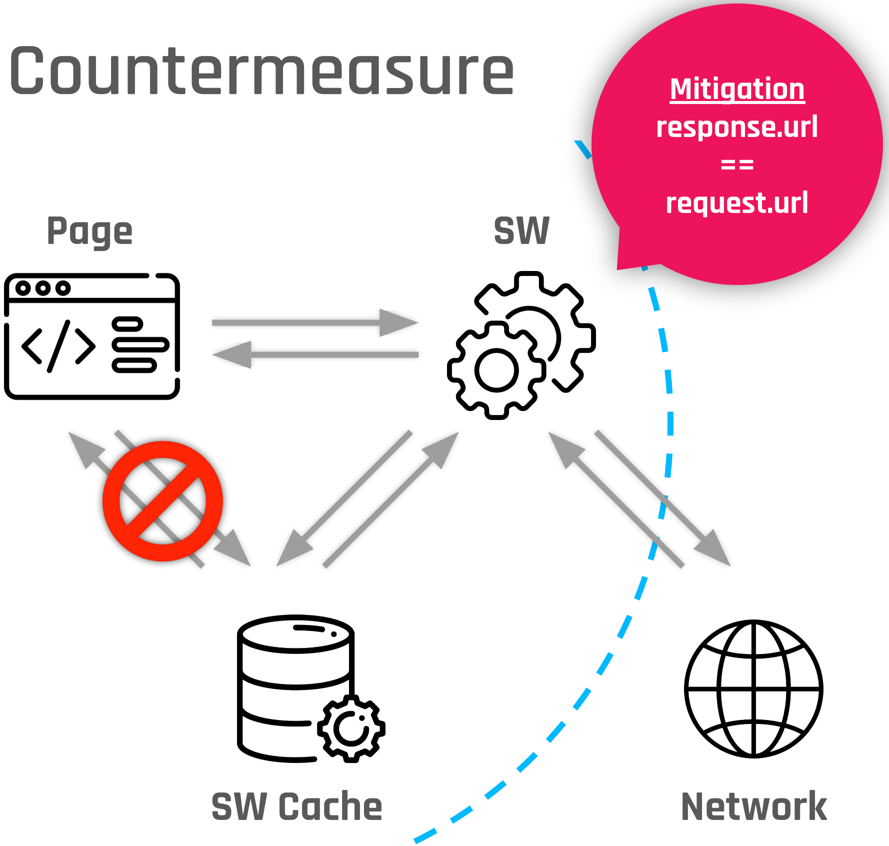
- Crawled Tranco top **150K sites**, visited **>4M pages** (June '20)
  - **6,709** sites use **Service Workers (4.6%)**
  - **3,436** sites use **Service Workers** + **Cache API (51.2%)**
  - **Broken or missing CSP in 95.8% of sites using SW + Cache API**
    (Potentially vulnerable to our attack if a XSS is found in a page of the site)

- Automated vulnerability testing
  - **2,769** (**65%**) sites **blindly execute** a JS payload we added to cached content (HTML or scripts)
  - 2,040 sites cache HTML (38% executes)
  - 2,148 sites cache JS (75% executes)

Cached Policies



Countermeasure



Straightforward solution

- **Restrict Cache API to SW**
- Compatibility issues with existing sites:
  - **~6%** of the sites using the Cache API, access the cache from a script
  - Identified legitimate patterns

Compatible solution

- **Restrict Cache API to SW by default**
- **Custom header** or integration with **DocumentPolicy** to **relax the protection**

Conclusion

- **Powerful attack** against **Service Workers** on the design of the **Cache API**

- **PITM-like capabilities** that couldn't be achieved by a persistent client-side XSS

- Strong, but **realistic**, **threat model**
  - **XSS** still **widespread** (35.6% of the Google Vulnerability Reward Program payout in 2018 ~ 1.2M $)*
  - **CSP** often **misconfigured** (~95%)
  - **Large scale assessment** (150K sites) + successful **automated testing** (65%)

- Proposed a **backward-compatible redesign** of the **Cache API** that would have an **immediate security benefit** for the large majority of websites

* **Artur Janc. Baby steps towards the precipice** https://www.arturjanc.com/usenix2019/

Demos, PoCs, Extension, Paper ↴

https://swcacheattack.secpriv.wien/

Icons from https://www.flaticon.com

Thank you!

https://swcacheattack.secpriv.wien/

Q+A?

Marco Squarcina (TU Wien)

✉ marco.squarcina@tuwien.ac.at

🐦 @blueminimal