MASTER OF WEB PUPPETS

Abusing Web Browsers for Persistent and Stealthy Computation

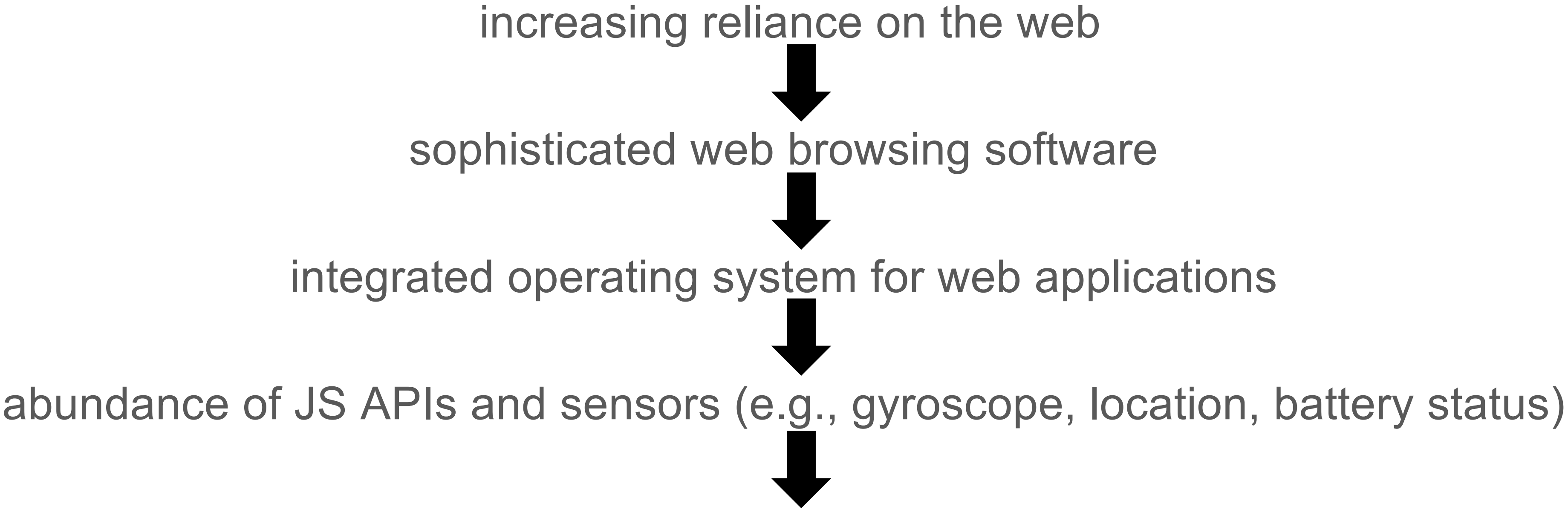
Panagiotis Papadopoulos

Panagiotis Ilia, Michalis Polychronakis, Evangelos Markatos,
Sotiris Ioannidis, Giorgos Vasiliadis

Stony Brook University

FORTH
Foundation for Research & Technology - Hellas

Nowadays...



increasing reliance on the web

sophisticated web browsing software

integrated operating system for web applications

abundance of JS APIs and sensors (e.g., gyroscope, location, battery status)

publishers transfer parts of the critical computations on the user side

(minimizes latency, better user experience and usability, scalability of the service)

Nowadays...

in-browser running malicious code (e.g., cryptojacking)

⬆

users remain oblivious to the performed operations

⬆

publishers are considered by default trusted to run *any* JS code
on the user-side without restrictions (even from third-parties)

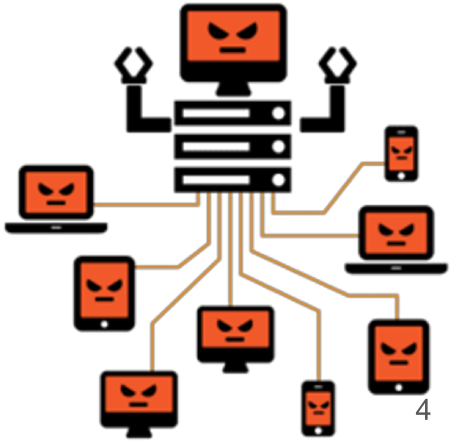
⬆

publishers transfer parts of the critical computations on the user side
(minimize latency, better user experience and usability, scalability of the service)

Threat Model

Publishers can infect users:

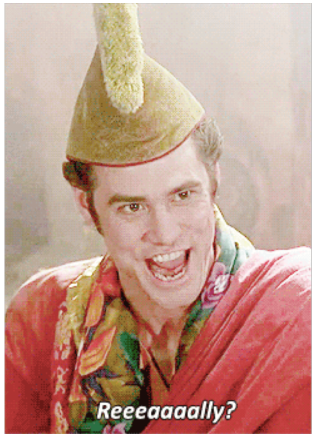
- Intentionally:
  - malicious or “shady” website serves directly the malicious payload to visitors
- Unintentionally:
  - hijacked/compromised website
  - website that includes a third party library which becomes compromised
  - malicious third-party content that it dynamic loaded in iframes*
    (e.g., through real-time ad auctions).



* Browser as Botnet, or the Coming War on Your Web Browser
https://medium.com/@brannondorsey/browser-as-botnet-or-the-coming-war-on-your-web-browser-be920c4f718

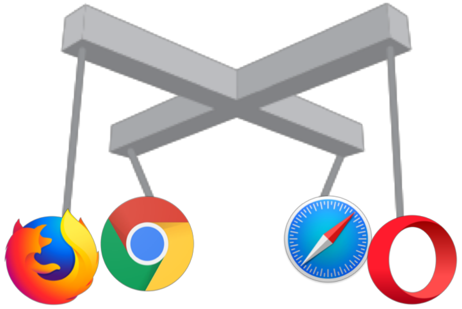
On the bright side...

malicious JS execution is constrained chronologically to the lifetime of the browser window or tab



This is not the case nowadays...

to demonstrate that we present **MarioNet**



MarioNet

- a system that enables a remote attacker to control users' browsers and hijack device resources
- infected browser joins a centrally orchestrated botnet which can launch a wide variety of distributed attacks
- *persistent* and *stealthy: attackers* continue having control of the victim's browser *even after the user browses away* from the website
- leverages only off-the-shelf technologies provided by HTML5

MarioNet: Basic ingredients

- Service Worker:
  - non-blocking (i.e., fully asynchronous) module,
  - resides between the webpage and the publisher’s web server:
  - once registered and activated, runs in a separate thread in the background (no DOM access)
  - can intercept and handle network requests
    (e.g., is used for programmatically managing the caching of responses)

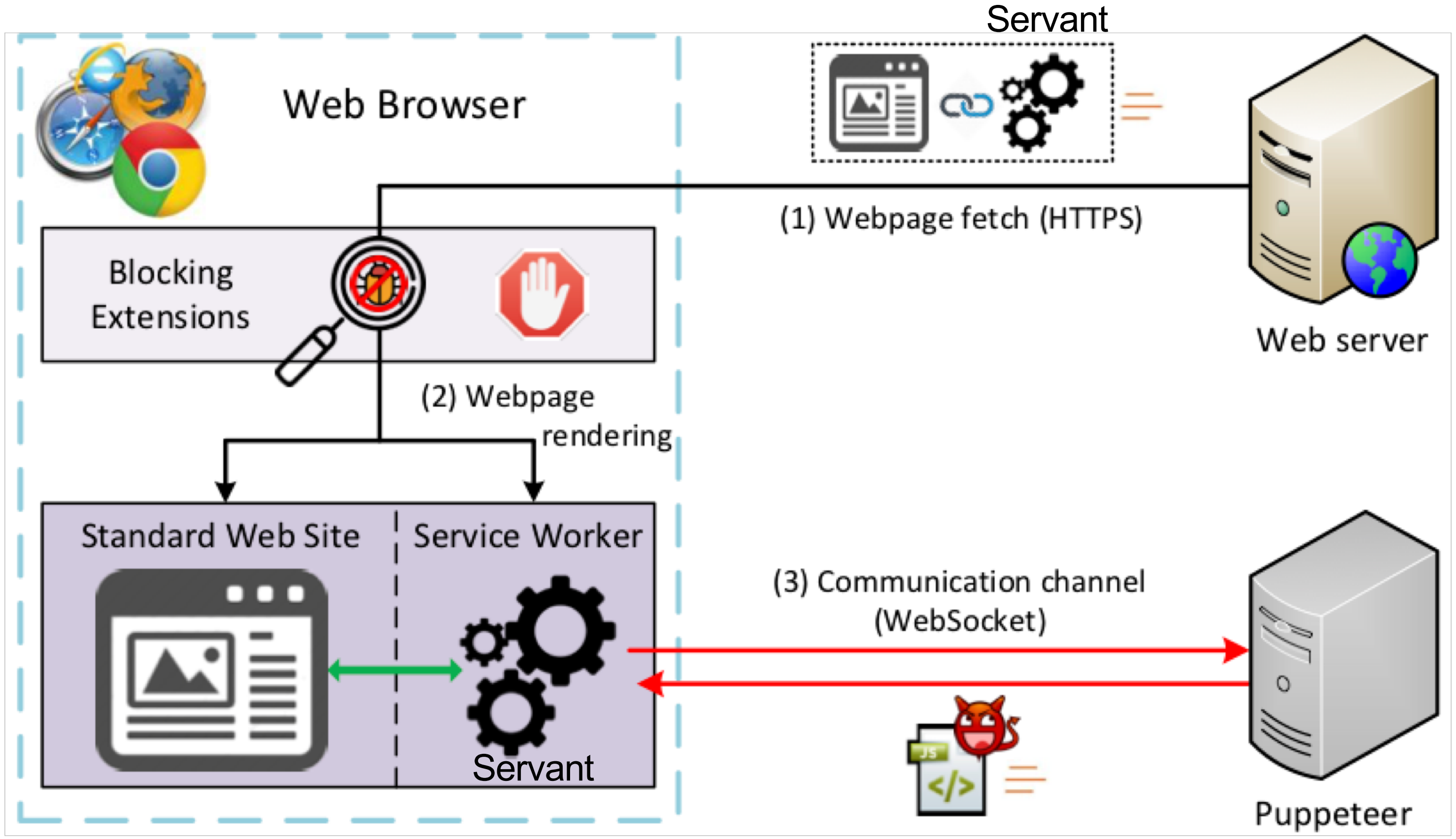
- WebSocket:
  - persistent full-duplex communication channel over a single TCP connection.

- Depending on the attack scenario:
  - WebRTC, high-resolution performance timers, Cross-Origin Resource Sharing (CORS)

MarioNet: Key roles

1. **Distributor:**
   - ○ a website under the attacker’s control that delivers to users along with the regular content of the webpage, the Servant component
2. **Servant:**
   - ○ the in-browser component embedded in a Service Worker
   - ○ runs in a separate process: continues to operate even after its parent tab closes
   - ○ establishes a connection with Puppeteer for heartbeats and receiving malicious tasks
3. **Puppeteer:**
   - ○ the remote C&C component that sends tasks to the Servant to be executed
   - ○ sets the attack’s target, orchestrates the botnet

MarioNet: Overview



Basic Characteristics

1. ***Isolation:***
   - MarioNet's operations are independent from the browsing session's thread/process
   - heavyweight malicious computations cannot affect tab's functionality

2. ***Persistence:***
   - MarioNet’s operations are detached from any ephemeral browser tabs
   - browser remains under attacker’s control for longer than a website visit (<1 min*)

3. ***Stealthiness:***
   - Servant-Puppeteer communication channel is not-detectable by browser extensions**
   - operations are throttled based on system's recourse utilization (not-detectable by user)

***How long do users stay on web pages?** https://www.nngroup.com/articles/how-long-do-users-stay-on-web-pages/

** Bashir et al., **How Tracking Companies Circumvented Ad Blockers Using WebSockets**

Attack Scenarios

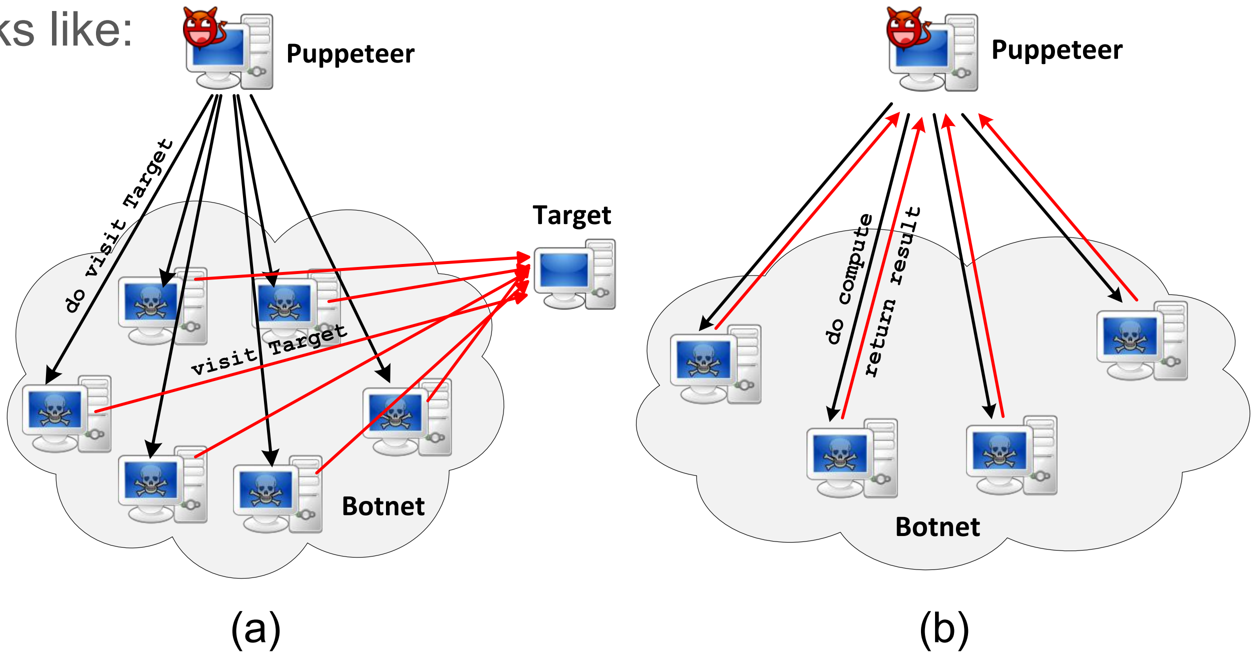
After infection Puppeteer can instrument infected browsers to perform attacks like:

a) DDoS

b) cryptocurrency mining

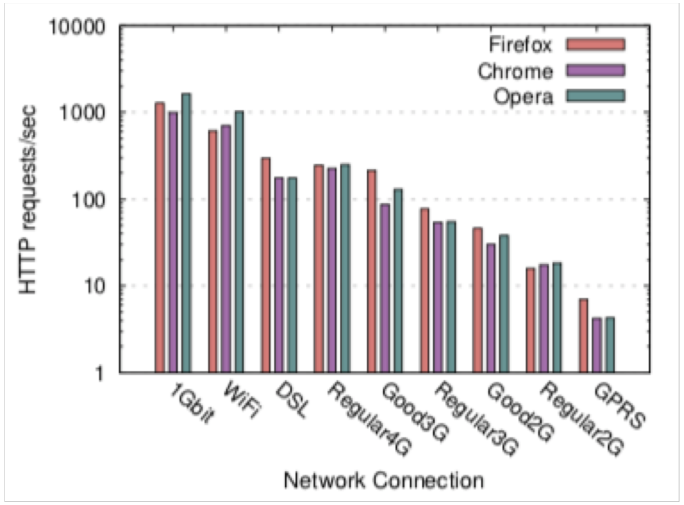
c) distributed password cracking

d) illicit file hosting or anonymized communications



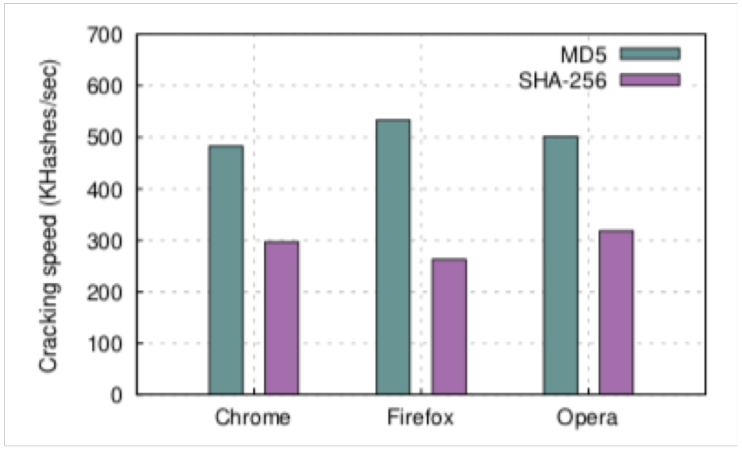
Performance Evaluation

- *Abuse of network resources: DDoS*



Rate of asynchronous outgoing HTTP requests for different browsers and network connections in the DDoS attack scenario. An orchestrated DDoS attack in MarioNet can achieve rates of up to 1632 reqs/sec per infected device.

- *Abuse of computation power: Distributed 10-digit Password Cracking*



Cracking speed of different browsers in the distributed password-cracking scenario. A single infected browser can bruteforce around 500K MD5 hashes/sec or 300K SHA-256 hashes/sec.

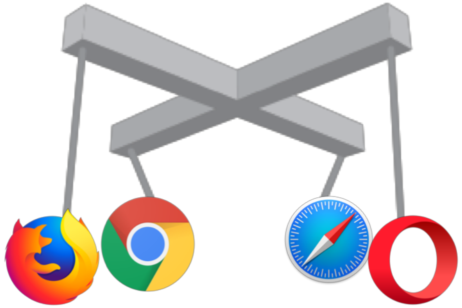
Countermeasures

- *Restricting or Disabling Service Workers:*
  - *breaks important functionality of apps like Google Docs, Gmail, LinkedIn, Whatsapp web client (e.g., periodic background synchronization, push notifications, caching, message relaying across pages, offline fallback, user-side load balancing)*
- *Whitelists/Blacklists:*
  - Service Workers will be blocked, unless the domain of origin is whitelisted.
- *Click to Activate:*
  - require user's permission before registering a service worker (like Push Notifications)
- *Host-based approaches:*
  - *Signature-based Detection*
  - *Behavioral Analysis and Anomaly Detection*

Conclusion

- Malicious code execution is not constrained to the lifetime of the browser session

- We present ***MarioNet***:
  - Ingredients: Service Workers, WebRTC, WebSockets, CORS
  - a multi-attack framework
  - allows persistent and stealthy bot operation through web browsers
  - Malicious payload is not attached to any browsing session and thus withstands tab crash/closing.



- We launch and evaluate various possible attack scenarios on top of MarioNet: (e.g., DDoS, Distributed password cracking, cryptomining)

Persistency and fixes (update)

1. Issue and fixes were made public in November 2018[1,2]
2. first fixes were already integrated since end of July
   - restricting the number of events (e.g., fetch, push, sync, etc.) SWs can receive
3. Currently, SW gets only one event and then gets terminated
4. SW stays alive only for ~1min after the user navigates away from the website
   - limits the lifetime of an infected user.

More info: https://www.ics.forth.gr/dcs/index_main.php?l=e&c=735

[1] https://bugs.chromium.org/p/chromium/issues/detail?id=805496&can=1&q=Type%3DBug-Security%20serviceworker&sort=-modified&colspec=ID%20Pri%20M%20Stars%20ReleaseBlock%20Component%20Status%20Owner%20Summary%20OS%20Modified
[2] https://bugzilla.mozilla.org/show_bug.cgi?id=1432846

MASTER OF WEB PUPPETS

Abusing Web Browsers for Persistent and Stealthy Computation

Panagiotis Papadopoulos