CS590-SWS/527 Software Security

Browser Security

Asst. Prof. Mathias Payer

Department of Computer Science
Purdue University

TA: Kyriakos Ispoglou

https://nebelwelt.net/teaching/16-527-SoftSec/

Spring 2016

Table of Contents

Basics

- Uniform Resource Locators (URLs) allow global identification of retrievable documents, specifying both the protocol and location of the resource.
- Location is broken into hostname, port, path, query (a sequence of variable=value statements with & as delimeter), and fragment, e.g.,
  `protocol://hostname:port/path/?query#fragment`
- The Hyper Text Transfer Protocol (HTTP) allows a host to fetch URLs from a remote server. HTTP is a simple stateless protocol where a client requests a page (including some auxiliary headers). The response contains the HTTP version, status code, reason, headers, length, and data in a simple human readable format.

Browsers

- Modern browsers have evolved into complex applications that retrieve and display html documents retrieved over http.
- The browser process handles entering and displaying of the URI. Upon receiving a new display request the browser process asks the browser engine if the location is cached; if not, it requests the page over http.
- The renderer process takes web page and renders it.
- A window/frame uses a basic execution model that loops between content loading, rendering, and reacting to events (user actions, rendering, or timing).

Rendering

- Rendering is not that simple.
- A bunch of different parsers handle data in different formats.
- There's HTML, JavaScript, CSS, Images, and many more.
- All parsers and the JavaScript engine that allows dynamic code to be executed, update the Document Object Model (DOM) tree which is then "rendered".

Table of Contents

Security goals

- A tab cannot compromise data on the system.
- A tab cannot compromise data in other tabs.
- A tab cannot hijack and control other tabs.

Different threat models

- Web attacker: may control a server and/or certificate. User actively visits the page (or app or ad).
- Network attacker: passive eavesdropper or active man in the middle (MITM), or ARP/DNS poisoning attack.
- OS attacker: exploit browser and control execution.

Table of Contents

Browser isolation

- The browser isolates fault domains through processes.
- The browser process controls a GPU process, a sandbox for each renderer, and a set of sandboxes for different plugins.
- Each sandbox is isolated from the system and other sandboxes.
- Each sandbox is reduced to the least privilege.
- On Linux, chrome uses `seccomp` secured processes to enforce isolation and least privilege.

Table of Contents

Heap spraying

- Browser sandboxes are very constrained but an attacker may execute arbitrary code through JavaScript.
- Vulnerabilities like use-after-free are often imprecise due to variance in the heap (i.e., where objects are allocated).
- Idea: spray a large amount of the same object across the heap, allocating many objects. Only one object will then have to be hit in the exploit.
- Heap spraying conceptually solves the same problem as NOP slides.
- How would you detect heap spraying?

Information leaks

- Browsers have several side channels that allow an attacker to identify "loaded" resources.
- Web cache-based attacks: image/script load times
- Identification: supported fonts, plugin versions, sites visited
- System cache-based attacks: rowhammer, covert channels.

Table of Contents

Summary

- Browsers leverage HTTP and URLs to retrieve HTML documents.
- Browsers run individual webpages at least privileges in isolation.
- An attacker may control the content of the website at different levels.
- Vulnerabilities in the browser allow an attacker to escape and escalate privileges.
- Two examples of attack vectors are heap spraying and side channels.

Questions?

?