

Network and Web Security

Browser fingerprinting

Dr Sergio Maffeis

Department of Computing

Course web page: <https://331.cybersec.fun>

Network & browser fingerprinting

- Network fingerprinting
 - Detect configuration information to identify a system component
 - An important part of the pentesting intelligence gathering phase
 - Which host is the DNS server?
 - What OS is running on each host?
 - What services are running on the open ports of a host?
- Browser fingerprinting
 - Recognize the same browser instance across website visits
 - *That* particular browser binary, on *that* particular device
 - Goals
 - Authentication
 - Google will send you an email if you log in from a device it hasn't seen before
 - Authorization
 - Session tokens may include hash of browser fingerprint to prevent session hijacking
 - Access control
 - Network or service access may be restricted to a particular known device
 - Tracking, deanonymisation
 - If two website visits show same fingerprint then user is likely to be the same

Passive fingerprinting

- The servers receives an HTTP request
 - TCP/IP parameters, including IP address
 - A number of HTTP headers
- Some headers directly reveal information
 - User-Agent, Accept, Accept-Language, Accept-Encoding
- Identity and order of headers also matter
- Accessing the same website, same machine, same OS on:
 - **Chrome**

GET / HTTP/1.1

Host: www.theguardian.com

Connection: keep-alive

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8

Upgrade-Insecure-Requests: 1

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/48.0.2564.116 Safari/537.36

Accept-Encoding: gzip, deflate, sdch

Accept-Language: en-US,en;q=0.8

- **Firefox**

GET / HTTP/1.1

Host: www.theguardian.com

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:44.0) Gecko/20100101 Firefox/44.0

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Language: en-GB,en;q=0.5

Accept-Encoding: gzip, deflate

Connection: keep-alive

Active fingerprinting

- Popular techniques using JavaScript and plugins
 - Installed fonts
 - Works best in Java/Flash
 - Possible in JavaScript via tricks
 - Installed plugins
 - Browser type
 - Also reported by HTTP headers
 - Time-zone
 - Screen resolution and color depth
 - JavaScript engine performance
 - JavaScript engine conformance to standard
 - Supported features, presence of browser-specific objects and behaviours
 - GPU or graphic drivers
 - Exposed by differences in rendering elements via HTML5 <canvas> element
- Concrete example:
<https://github.com/Valve/fingerprintjs2>



Commercial fingerprinting

- Some private companies provide fingerprinting services, mostly to marketers
 - Fore example: Bluecava, Iovation ReputationManager and ThreatMetrix
 - They do it “openly”, yet the user does not know, or notice
 - Excellent source of examples of tracking and fingerprinting code
- To find out more: *Exploring the Ecosystem of Web-based Device Fingerprinting*

Fingerprinting Category	Panopticlick	BlueCava	Iovation ReputationManager	ThreatMetrix
<i>Browser customizations</i>	Plugin enumeration(JS) Mime-type enumeration(JS) ActiveX + 8 CLSIDs(JS)	Plugin enumeration(JS) ActiveX + 53 CLSIDs(JS) Google Gears Detection(JS)		Plugin enumeration(JS) Mime-type enumeration(JS) ActiveX + 6 CLSIDs(JS) Flash Manufacturer(FLASH)
<i>Browser-level user configurations</i>	Cookies enabled(HTTP) Timezone(JS) Flash enabled(JS)	System/Browser/User Language(JS) Timezone(JS) Flash enabled(JS) Do-Not-Track User Choice(JS) MSIE Security Policy(JS)	Browser Language(HTTP, JS) Timezone(JS) Flash enabled(JS) Date & time(JS) Proxy Detection(FLASH)	Browser Language(FLASH) Timezone(JS, FLASH) Flash enabled(JS) Proxy Detection(FLASH)
<i>Browser family & version</i>	User-agent(HTTP) ACCEPT-Header(HTTP) Partial S.Cookie test(JS)	User-agent(JS) Math constants(JS) AJAX Implementation(JS)	User-agent(HTTP, JS)	User-agent(JS)
<i>Operating System & Applications</i>	User-agent(HTTP) Font Detection(FLASH, JAVA)	User-agent(JS) Font Detection(JS, FLASH) Windows Registry(SFP)	User-agent(HTTP, JS) Windows Registry(SFP) MSIE Product key(SFP)	User-agent(JS) Font Detection(FLASH) OS+Kernel version(FLASH)
<i>Hardware & Network</i>	Screen Resolution(JS)	Screen Resolution(JS) Driver Enumeration(SFP) IP Address(HTTP) TCP/IP Parameters(SFP)	Screen Resolution(JS) Device Identifiers(SFP) TCP/IP Parameters(SFP)	Screen Resolution(JS, FLASH)

(Nikiforakis et al., 2013)

Key trade offs

- Passive vs Active fingerprinting
 - Passive fingerprinting
 - Cannot be detected or prevented
 - Does not affect target
 - Can gather only the information exposed by the target
 - Active fingerprinting
 - May be detected and prevented
 - Can disrupt the target
 - Can probe deeper into the target and reveal more precise information
- Precision vs stability of a fingerprint
 - Precision: a good fingerprint should be different for any 2 devices
 - Stability: a useful fingerprint for a particular device should not change much over time
 - Embedding more attributes in a fingerprint is likely to increase precision but decrease stability
- Building a *fingerprint*
 - Conceptually, a fingerprint is a set of attributes
 - Once fingerprint is known, hash can be used to reduce network traffic (if fingerprint is stable)
 - What attributes to use in the fingerprint? What data for each attribute?
 - Goal: optimise the precision-stability trade-off
 - Domain knowledge may help defining rules to increase stability
 - Example: if minor version of browser or OS increases, may consider same device with some degree of confidence

Fingerprinting statistics

- Demo websites collect and fingerprints and analyse data
 - <https://panopticklick.eff.org/> the first one to do it, now branched also on tracking
 - <https://amiunique.org/> most advanced on fingerprinting, lots of interesting graphs
- Methodology
 - Run fingerprinting routines on visiting browser and report detected attributes
 - Compute entropy of each value with respect to observed samples
 - As a proxy for precision
 - A summary “entropy” statistics supposedly representing uniqueness of your browser
 - Good to raise awareness of problems and techniques
 - Doubts remain about validity of the estimation
 - Data is biased: visitors tend to be privacy-conscious users not representative of broader internet population
 - Most features in fact are highly correlated
 - Results on entropy/precision are flattered by lack of stability study

Countermeasures

- Different motivations to fight fingerprinting
 - Good guys want to preserve privacy
 - Bad guys want to commit fraud or crime
- Tools to fight back
 - Browser extensions, browser options, network proxies/firewalls
- Nuclear option
 - Leak minimal configuration information
 - Blacklist known fingerprinters: Firefox now does that automatically
 - Rewrite HTTP requests to hide sensitive information
 - Disable plugins and JavaScript
 - Drawbacks: degradation of user experience, some sites will just break
- Mimic a target
 - Spoof information to report the fingerprint of a system target of impersonation
 - Constrained by difference between attacker and user device and context (device, IP, etc)
- Hide in the crowd
 - Spoof information to report a very common fingerprint compatible with user device
 - Fingerprint coincides with that of many unrelated users, privacy is protected
- Destabilise fingerprint
 - Spoof information to present a different fingerprint at each visit to same fingerprinter
 - Fingerprint may often result to be unique, but will also be highly unstable, providing privacy

Anti-fingerprinting examples

Attack point	Entropy	Defence	How
User-agent string	High	spoof	Modify BOM navigator object
Plugins	High	spoof/randomisation/disable flash	Modify BOM navigator object
Fonts	High	spoof/randomisation/disable flash	modify offsetheight/width
HTTP Accept header	High	spoof	Chrome.webRequest API
Screen resolution	Medium/low	spoof	Modify BOM screen object
"DoNotTrack"	Medium/low	spoof	Modify BOM navigator object
Language	Medium/low	spoof	Modify BOM navigator object

```
Object.defineProperty(HTMLElement.prototype, 'offsetWidth', {  
  ...  
  get: function(){  
    return this.clientWidth + (getRandomInt(-5, 5)/100)*this.clientWidth;  
  },  
  set: function(newval){  
    this.setAttribute('offsetWidth', newval);  
  }  
});
```

Anti-fingerprinting solutions

- Many solutions claim to stop fingerprinting
 - Very difficult to evaluate
 - Usability concerns
 - Still active area of research

Countermeasure	Problem of display	Problem of functionality	Difficult to use
Tor	√	√	√
RubberGlove	√	√	√
CanvasFingerprintBlock	-	-	-
Canvas Fingerprinting	-	-	-
FireGloves	√	√	√
FP-Block	-	√	-
Stop Fingerprinting	-	√	-

(Luangmaneeerote et al., 2016)

Countermeasure	Object JavaScript (navigator, screen)	List of fonts	List of plugins	Canvas
Tor	√	*	√	-
RubberGlove	√	-	√	-
Chameleon	≈	-	-	≈
CanvasFingerprintBlock	-	-	-	≈
ChromeDust	-	-	-	-
StopFingerprinting	-	-	-	-
Canvas Fingerprinting blocker	-	-	-	√
FireGloves	√	√	√	-
FP-Block	√	-	√	-
Stop Fingerprinting	-	√	√	-
UserAgent Switcher to Chrome	-	-	-	-

NWS - Browser fingerprinting

Counter-countermeasures

- Arms race between fingerprinter and “attacker”
- If anti-fingerprinting is a concern, *robustness* becomes another parameter to choose fingerprintable attributes
 - A *robust* attribute is one that is hard to spoof, that impacts usability, that can be cross-validated with other attributes
 - For example: network latency, IP, screen size, OS/browser/device agreement
- Attacker needs to avoid inconsistencies in reported data
 - Switch off plugins
 - They provide independent way to cross-check information and detect spoofing
 - Spoof HTTP headers consistently with DOM
 - `User-Agent` should be the same as `navigator.userAgent`
 - Avoid implausible configurations
 - Android browser with enormous screen size
 - Build database of acceptable configurations and select randomly across consistent options
- Attacker needs to avoid detection
 - Presence of anti-fingerprinting behaviour is in itself a fingerprintable feature
 - Fingerprinting libraries now include code to detect spoofing attempts
 - Defensive JS techniques may help avoid detection
- Attacker tries to detect activity of fingerprinting code
 - Avoid spoofing results to legitimate requests to improve user experience