Blocking Javascript

- Blocking of scripts used for ads & tracking is becoming more popular
- Mixed scripts combine ad/tracking and functional components
- Identify *functions* that are used for tracking and block those
  - Create “surrogate” versions
- Avoid blocking “gateway” functions that are used for network connections

Graph model

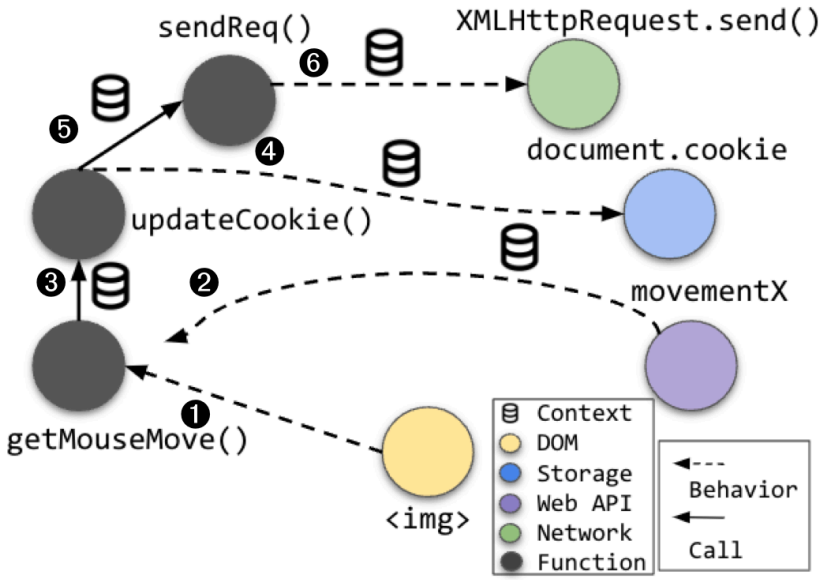

Figure 6: A simplified NoT.js's graph on mouse movement tracking.

Features	send Req()	update Cookie()	getMouse Move()
Number of requests sent	1	1	1
Is gateway function	1	0	0
Number of cookie (setter)	0	1	0
Number of web API (getter)	0	0	1
Number of arguments	1	1	0
Number of callee functions	1	1	0
Number of caller functions	0	1	1
Ascendant has cookie accesses	1	0	0
Descendant has cookie accesses	0	0	1
Ascendant has web API accesses	1	1	0

Machine Learning

- Labeling: function is tracking if:
  - it participates in call stack of a tracking request (as determined by blocklist)
  - It *does not* participate in a non-tracking request
- Use this to train a random forest classifier

Model	Section	Precision	Recall	F1 Score
NoT.js	Standard - 5.1	94.3%	98.0%	96.2%
NoT.js	Obfuscation - 5.3	93.5%	90.4%	91.9%
NoT.js	Coverage - 5.3	88.4%	95.7%	91.9%
WebGraph	Comparison - 5.4	49.3%	66.4%	56.5%
SugarCoat	Comparison - 5.4	23.0%	22.6%	22.8%

Table 4: NoT.js's precision, recall, F1-score in standard settings, enhanced coverage, obfuscation robustness, and comparison with existing tools.

Breakage

Category	WebGraph		SugarCoat		NoT.js	
	Minor	Major	Minor	Major	Minor	Major
Navigation	0%	6%	0%	0%	0%	0%
SSO	2%	2%	2%	0%	2%	0%
Appearance	4%	0%	0%	0%	4%	0%
Miscellaneous	4%	4%	4%	0%	4%	0%

Other Observations

- 32.1% of functions are tracking (?!)
- 13.4% of scripts are mixed, with 62.3% of websites having at least one mixed script
- Only 3.9% of functions are mixed (both tracking & non-tracking)
  - 0.8% of these functions must be blocked

Discussion

- How resilient is this approach? E.g., function names changes, mixed functions, adversarial ML
- Is the performance overhead practical?
- Does this approach introduce security vulnerabilities?
- Are false positives low enough? What is a usable level? Can it be user-tunable?
- Again, what is the blocking end game? Server-side tracking?
- Can this be used in other contexts, e.g., malicious scripts?

Wrap Up

- Other discussion points?
- What did you find surprising?
- Who really liked this paper? Really hated it?