Pool-Party

Exploiting Browser Resource Pools for Web Tracking

Peter Snyder – Brave Software
Soroush Karami – University of Illinois at Chicago
Arthur Edelstein – Brave Software
Benjamin Livshits – Imperial College London
Hamed Haddadi – Brave Software, Imperial College London

Pool-Party Attacks in a Slide

- Browsers (mostly) try to prevent cross-site tracking
- They partitioning resources by site (cookies, caches…)
- Many implementation-resources are not partitioned
- These can be exploited to enable cross-site track
- Previously known possible, this work shows they’re practical

Overview

- **Defining pool-party attacks**
  What they are, how they differ from other privacy attacks, etc
- **Pool-party attacks in popular browsers**
  Which browsers, which APIs, across which contexts
- **Measuring how practical pool-party attacks are**
  Making sure we’re only breaking bad stuff…
- **Discussion and conclusions**
  Fixes, other vectors, and more

Overview

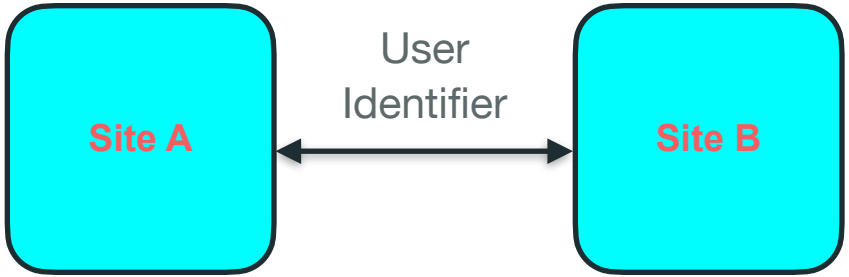
- **Defining pool-party attacks**
  What they are, how they differ from other privacy attacks, etc
- **Pool-party attacks in popular browsers**
  Which browsers, which APIs, across which contexts
- **Measuring how practical pool-party attacks are**
  Making sure we’re only breaking bad stuff…
- **Discussion and conclusions**
  Fixes, other vectors, and more

Defining pool-party attacks

- Category of covert channel in Web browsers,
- …across distinct contexts,
- …using resources that are limited and shared by those contexts

Defining pool-party attacks

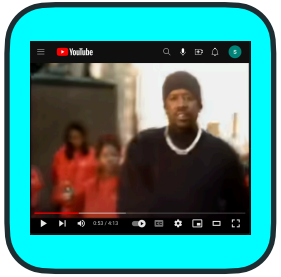
- Category of covert channel in Web browsers,
- …across distinct contexts,



- …using resources that are limited and shared by those contexts

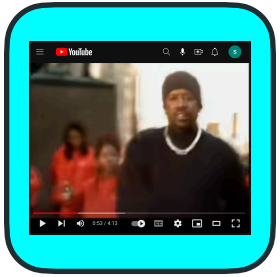
Toy pool-party attack

Site A



101011

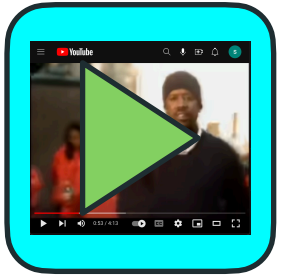
Site B



??????

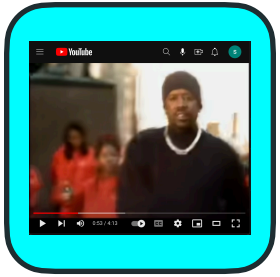
Toy pool-party attack

Site A



101011

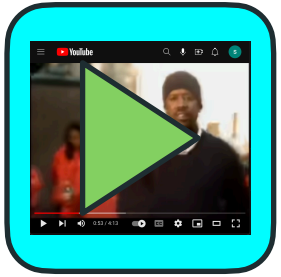
Site B



??????

Toy pool-party attack

Site A



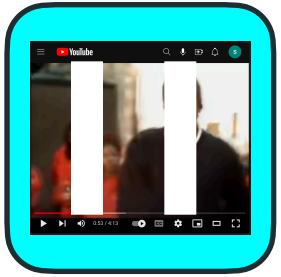
101011

Site B

1?????

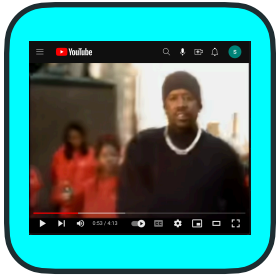
Toy pool-party attack

Site A



101011

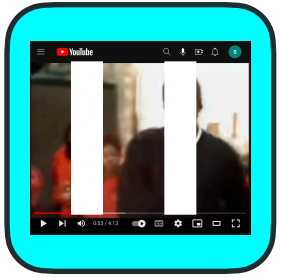
Site B



1?????

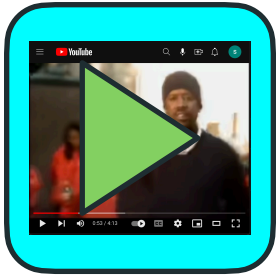
Toy pool-party attack

Site A



101011

Site B



10????

Generalizing properties of a pool-party attack

- **Resources are unpartitioned across contexts**
  Pool is shared across sites (or profiles, or storage clears)

- **Resource pool is limited to a predictable size**
  Sites can only consume resources to a known limit

- **Sites are otherwise unrestricted in consuming resources**
  No limit per context, other than global cap

- **Sites can learn when the global cap has been hit**
  Errors, communication failures, explicit messages, etc

Pool-party attacks in browsers?

- Network connection pools
- File handle pools
- Thread pools
- “In flight” request limits
- UI bottle necks / modal prompts

What makes a good attack?

- **Large pools**
  The more resources in the pool, the larger the size of each “packet”

- **Unpopular resources (features)**
  The less a feature is used on the Web, the less noise the covert channel

- **Quick to consume & release resources**
  Faster consume/release, larger bandwidth

Overview

- **Defining pool-party attacks**
  What they are, how they differ from other privacy attacks, etc
- **Pool-party attacks in popular browsers**
  Which browsers, which APIs, across which contexts
- **Measuring how practical pool-party attacks are**
  Making sure we're only breaking bad stuff…
- **Discussion and conclusions**
  Fixes, other vectors, and more

Finding pool-party vulnerabilities: Browsers

Browser	Engine	Version
Brave	Chromium	1.44.101
Chrome	Chromium	105.0.5195.125
Edge	Chromium	106.0.1370.42
Firefox	Gecko	105.0.1
Safari	WebKit	15.2
Tor Browser	Gecko	11.5.2

Finding pool-party vulnerabilities: APIs

- Manual process
- Source code review
- Consulting developers
- Standards / developer docs

Finding pool-party vulnerabilities: APIs

- **WebSockets**
  Persistent TCP-like interface for client-server communication

- **Web Workers**
  Sub-process-like API for running scripts outside of main event loop

- **Server-Sent Events**
  Server-push-like API for servers to notify pages of updates

Finding pool-party vulnerabilities

Browser	Contexts	WebSockets	Web Workers	SSE
Brave	Site	255	-	1,350
Chrome	Site	255	-	1,350
Edge	Site	255	-	1,350
Firefox	Site & Profile	200	512	-
Safari	Site	-	-	6
Tor Browser	Site	200	-	-

Finding pool-party vulnerabilities

Browser	Contexts	WebSockets	Web Workers	SSE
Brave	Site	255	-	1,350
Chrome	Site	255	-	1,350
Edge	Site	255	-	1,350
Firefox	**Site & Profile**	200	512	-
Safari	Site	-	-	6
Tor Browser	Site	200	-	-

Finding pool-party vulnerabilities

Browser	Contexts	WebSockets	Web Workers	SSE
Brave	Site	**255**	-	**1,350**
Chrome	Site	255	-	1,350
Edge	Site	255	-	1,350
Firefox	**Site & Profile**	200	512	-
Safari	Site	-	-	**6**
Tor Browser	Site	200	-	-

Overview

- **Defining pool-party attacks**
  What they are, how they differ from other privacy attacks, etc
- **Pool-party attacks in popular browsers**
  Which browsers, which APIs, across which contexts
- **Measuring how practical pool-party attacks are**
  Making sure we're only breaking bad stuff…
- **Discussion and conclusions**
  Fixes, other vectors, and more

Pool-party practicality

- **Bandwidth**
  How quickly can we transmit a user identifier across context boundaries

- **Consistency**
  How often does the attack succeed, given a stable, empty channel

- **Stability**
  How likely is it that the communication channel will be “clean”

Pool-party attack bandwidth & consistency

Browser	Attack Channel	Setup (s)	Send (s)	Total (s)	Success Rate
Brave	ServerSent Events	3.0	5.0	8.0	100%
Chrome	ServerSent Events	2.0	5.0	7.0	100%
Edge	ServerSent Events	2.0	5.0	7.0	100%
Chrome	WebSockets	0.1	0.5	0.6	100%
Edge	WebSockets	0.1	0.5	0.6	100%
Firefox	WebSockets	2.0	5.0	7.0	71%
Tor Browser	WebSockets	2.0	5.0	7.0	73%
Firefox	Web Workers	1.5	7.5	9.0	95%

Measurement are for a 35 bit identifier, over 100 measurements

Pool-party attack bandwidth & consistency

Browser	Attack Channel	Setup (s)	Send (s)	Total (s)	Success Rate
Brave	ServerSent Events	3.0	5.0	8.0	100%
Chrome	ServerSent Events	2.0	5.0	7.0	100%
Edge	ServerSent Events	2.0	5.0	7.0	100%
Chrome	WebSockets	0.1	0.5	0.6	100%
Edge	WebSockets	0.1	0.5	0.6	100%
Firefox	WebSockets	2.0	5.0	7.0	71%
Tor Browser	WebSockets	2.0	5.0	7.0	73%
Firefox	Web Workers	1.5	7.5	9.0	95%

Measurement are for a 35 bit identifier, over 100 measurements

Pool-party attack stability

Web API	% of page loads	% of desktop loads	% of mobile loads
Web Workers	12.34	12.29%	11.9%
WebSocket	9.55%	4.33%	3.72%
Server-Sent Events	0.79%	0.8%	0.06%

Figures from Chrome Platform Status telemetry (August 9, 2022)

Overview

- **Defining pool-party attacks**
  What they are, how they differ from other privacy attacks, etc
- **Pool-party attacks in popular browsers**
  Which browsers, which APIs, across which contexts
- **Measuring how practical pool-party attacks are**
  Making sure we're only breaking bad stuff…
- **Discussion and conclusions**
  Fixes, other vectors, and more

Pool-party discussion: additional vectors

- **Chromium & Gecko**
  - DNS resolver (64 simultaneous requests)
  - HTTP requests w/ HTTP proxy (32 requests)
  - OS pass through APIs (1 at a time)

- **WebKit**
  - Pre-fetch cache (64 hosts, GTK+ build only)
  - DNS resolver (8 simultaneous requests, GTK+ build only)

- **Almost certainly incomplete list…**

Pool-party discussion: defenses

- **Problem -> Unpartitioned and limited**

- **Solution 1: Partition (but maintain global cap)**
  - Each context gets its own allocation
  - Browsers: Brave

- **Solution 2: Removal global cap (but keep unpartitioned)**
  - No limit on availability
  - Browsers: Safari / WebKit

Take Aways

- Pool-party attacks exist(ed) in all browsers
- Practical and wide availability
- Probably more pool-party vulnerabilities
- Tracking on the Web is not a solved problem

Pete Snyder
pes@brave.com
@pes10k

More In the Paper

- Algorithmic details
- Comparison to other tracking techniques
- Measurement details