SECURING WEB 2.0 APPLICATIONS THROUGH REPLICATED EXECUTION

K. Vikram
Cornell University

Abhishek Prateek
IIT Delhi

Microsoft Research

Web 2.0 is Upon Us

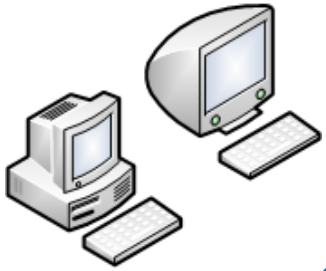
Web 1.0 → Web 2.0

Server-side

Advantage of the AJAX model:

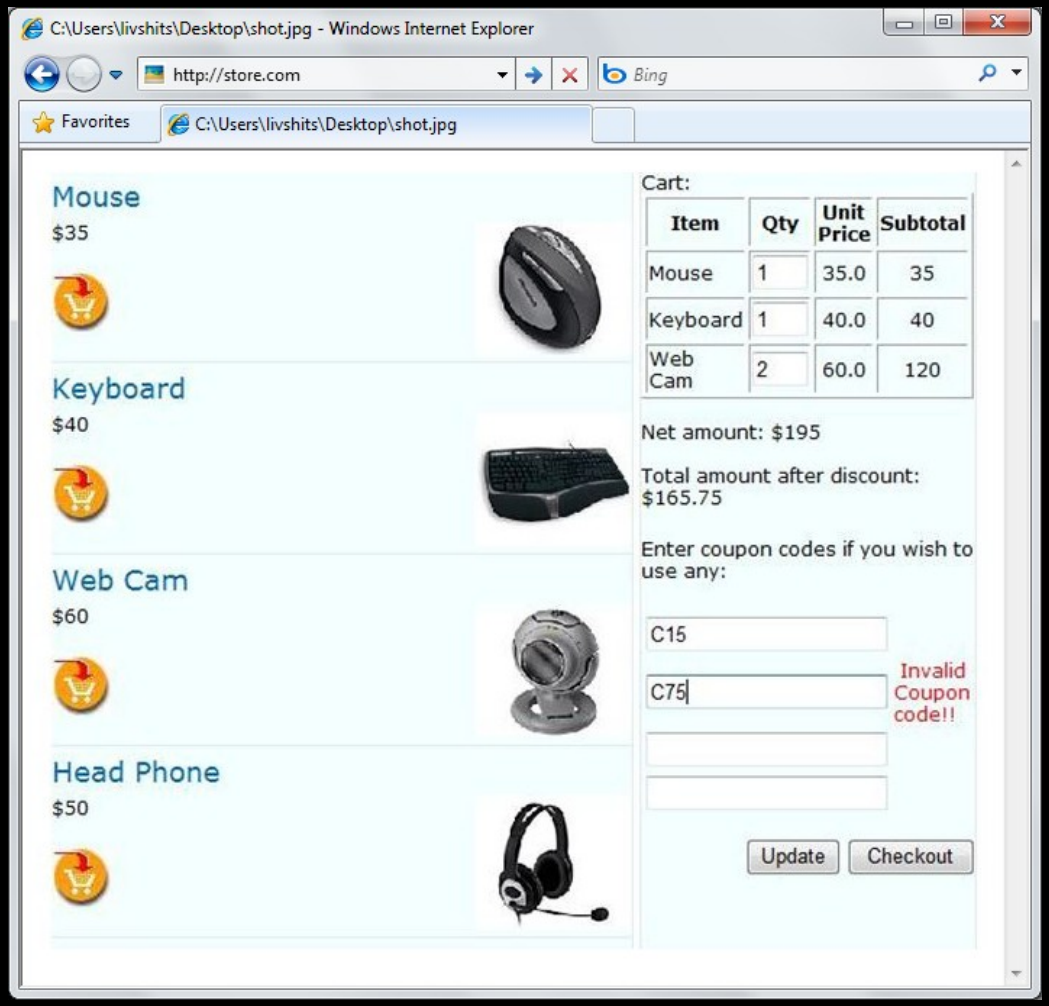
greater application responsiveness

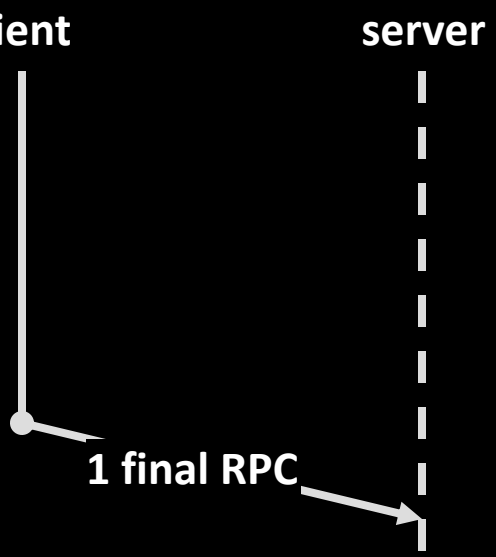
Client-side rendering



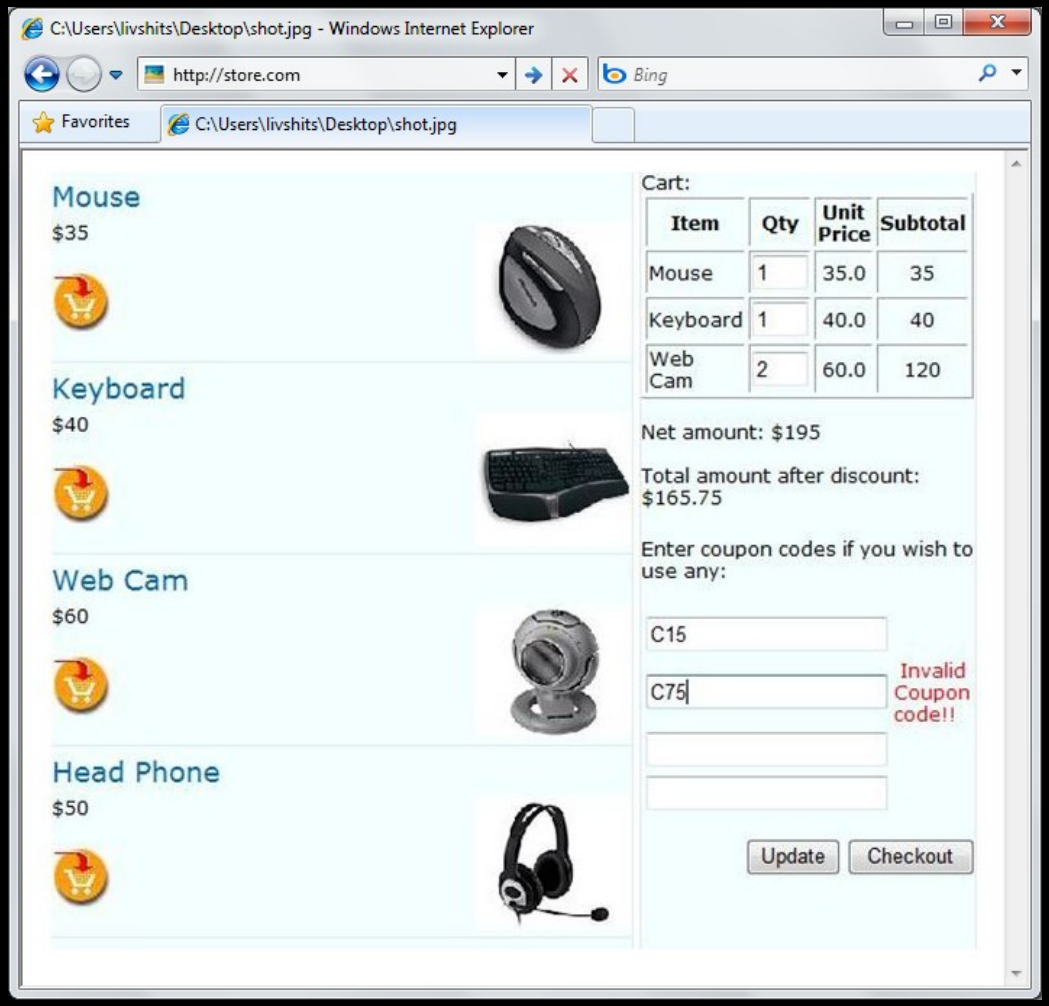
Motivation

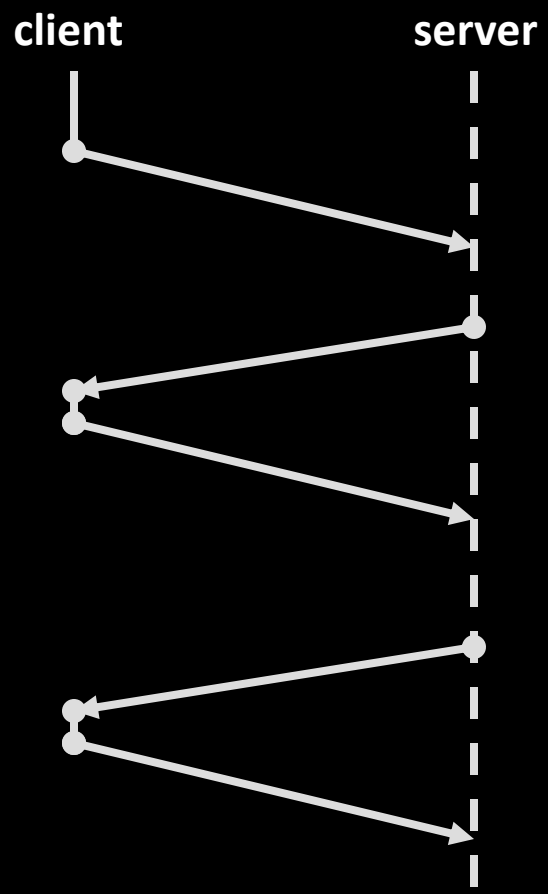
AJAX-based Shopping Cart (Fantasy)



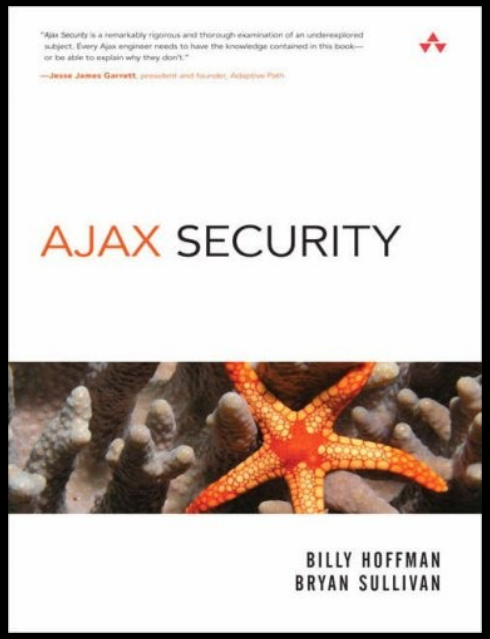


Shopping Cart (Reality)





Web Developer's Mantra



Thou shall not trust the client

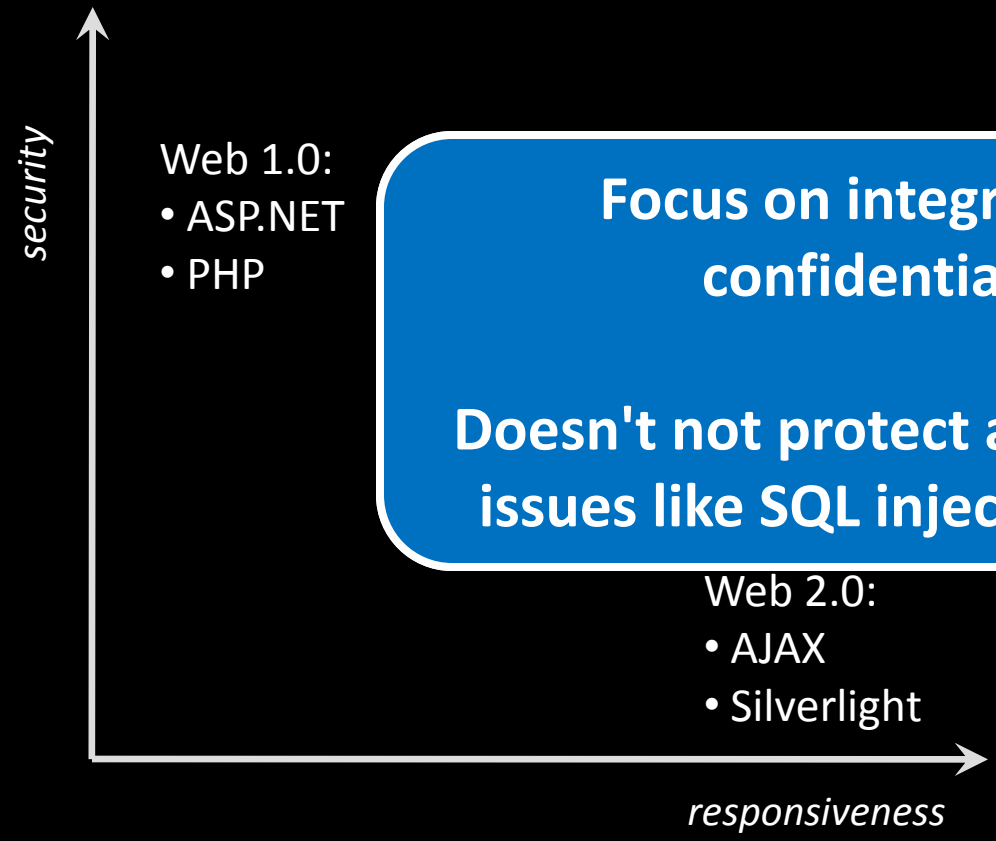
- No data integrity
- No code integrity

Tension Headaches

Move code to client for performance

Move code to the server for security

Security vs. Performance



Focus on integrity, not confidentiality

Doesn't not protect against other issues like SQL injection attacks

placing

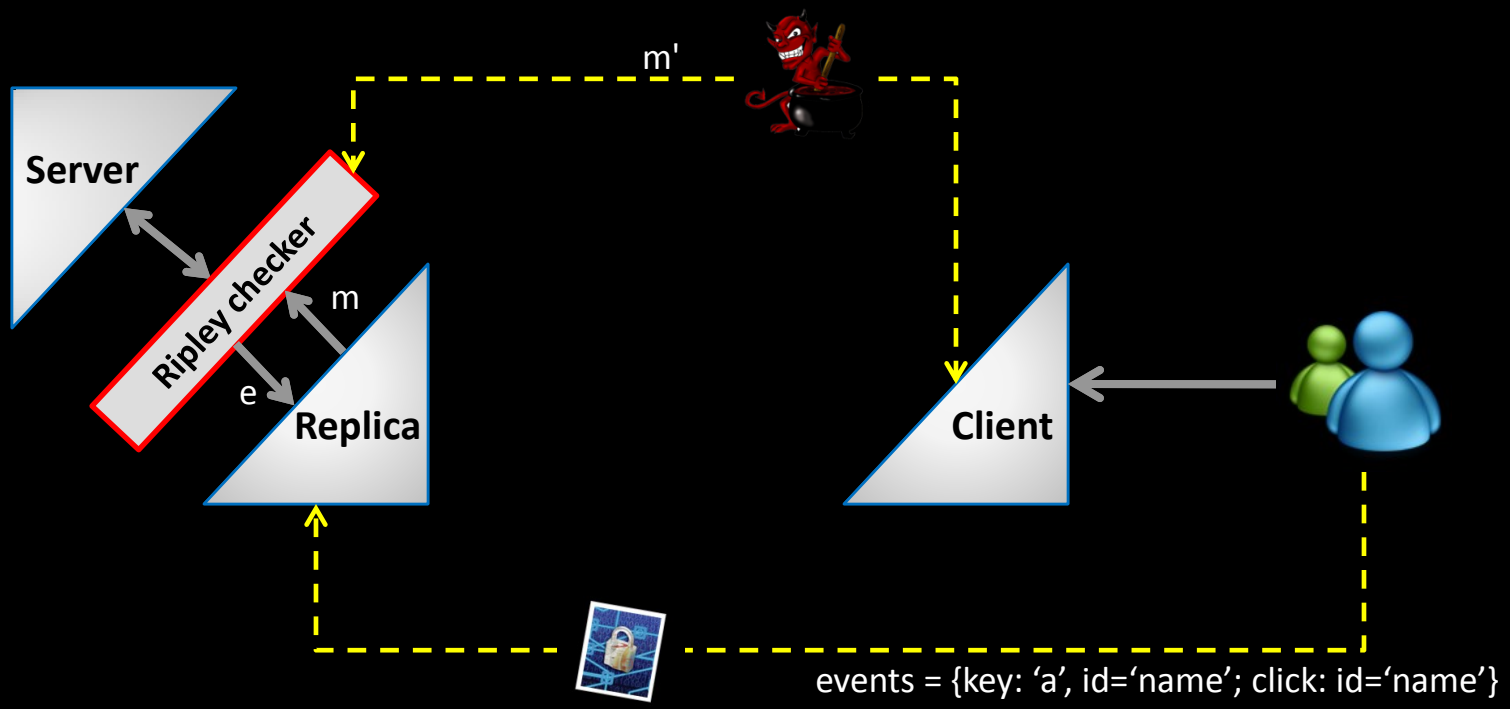
n the client

educe

nal integrity

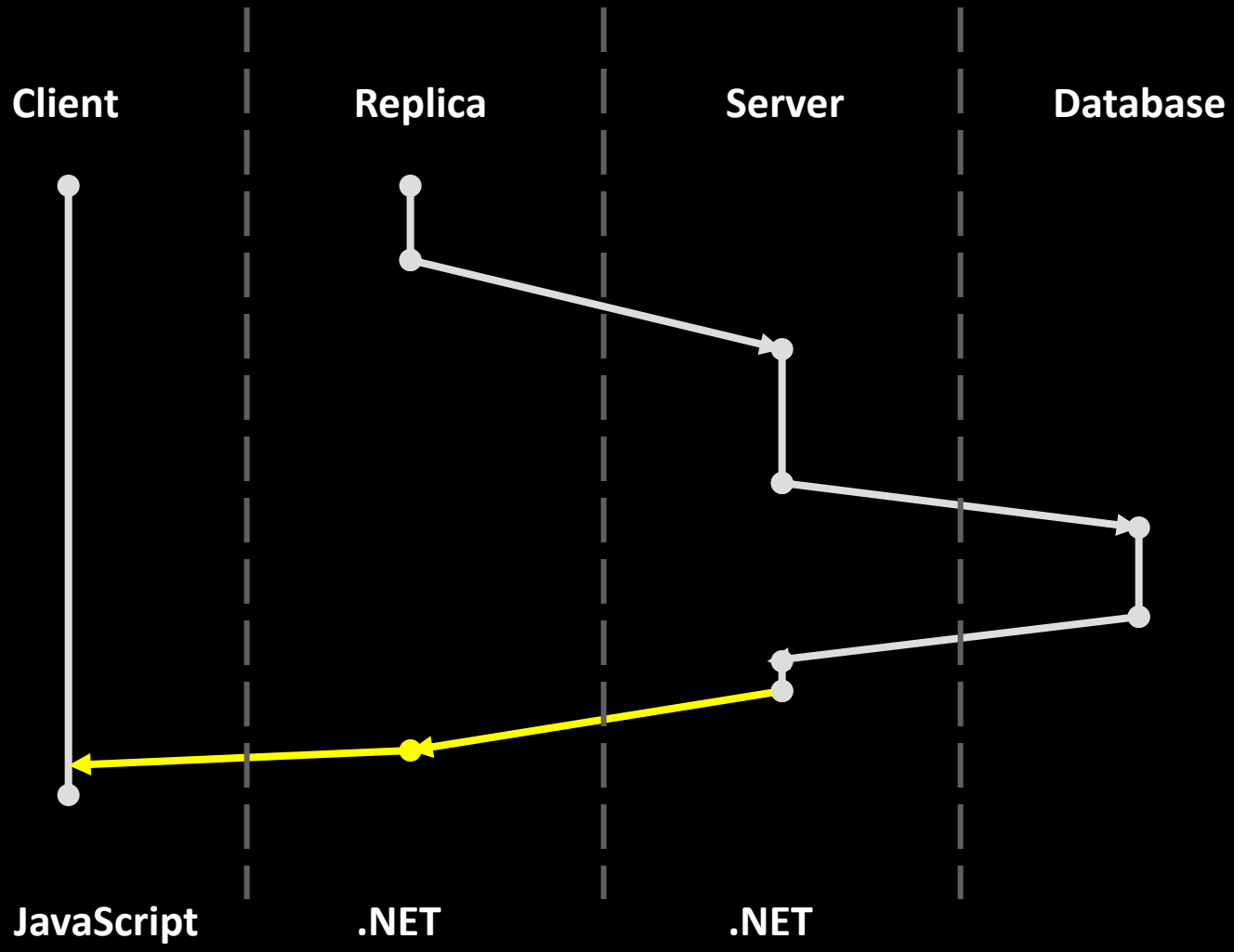
Architecture

Ripley Architecture



1. Keep a replica of the client code
2. Capture user events & transmit to server for replay
3. Compare server and client results

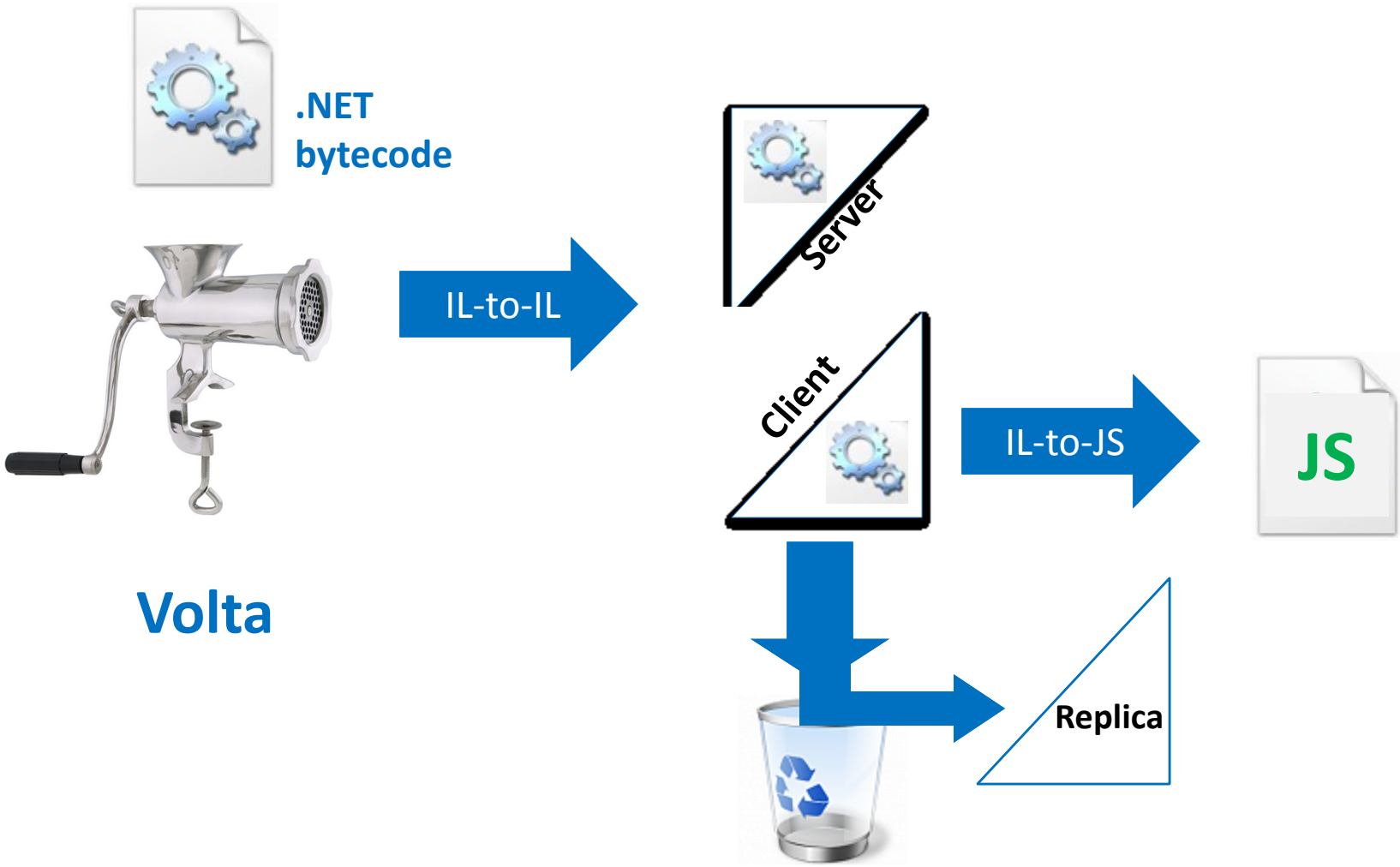
Zero-latency RPCs



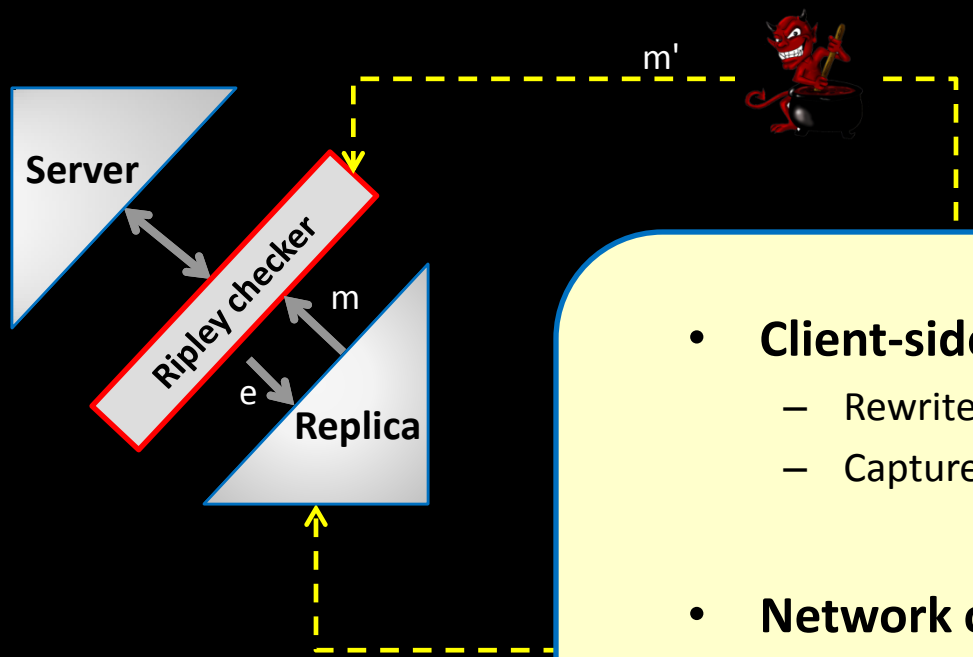
Seems Too Much Like Magic. Is this Feasible?

- Create deterministic replay system
  - How to we replicate JavaScript code?
  - Cross-browser differences?
  - Non-determinism?
- How do we scale it?
  - Replica overhead on server
  - *Hundreds* of concurrent replicas

The Volta Distributing Compiler Illustrated



Ripley Architecture

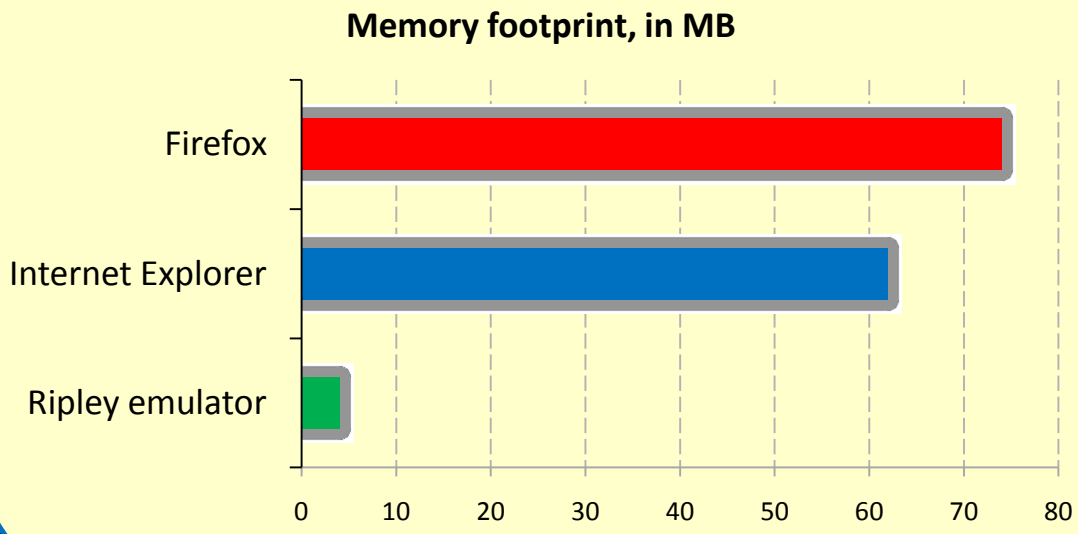


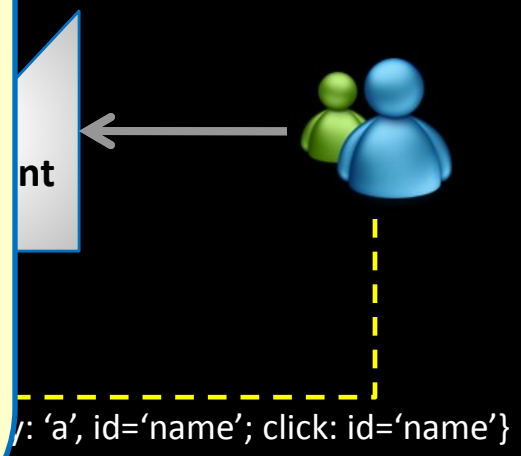
- **Client-side code instrumented**
  - Rewrite event handlers
  - Capture “default” events
- **Network overhead**
  - Buffer events for performance
  - Piggy-back on existing RPCs

1. Keep a replica of the cl
2. Capture user events & transmit to server for replay
3. Compare server and client results

Ripley Architecture

- Run replica in a Ripley emulator
- In .NET, not in JavaScript, 10-100x speed increase



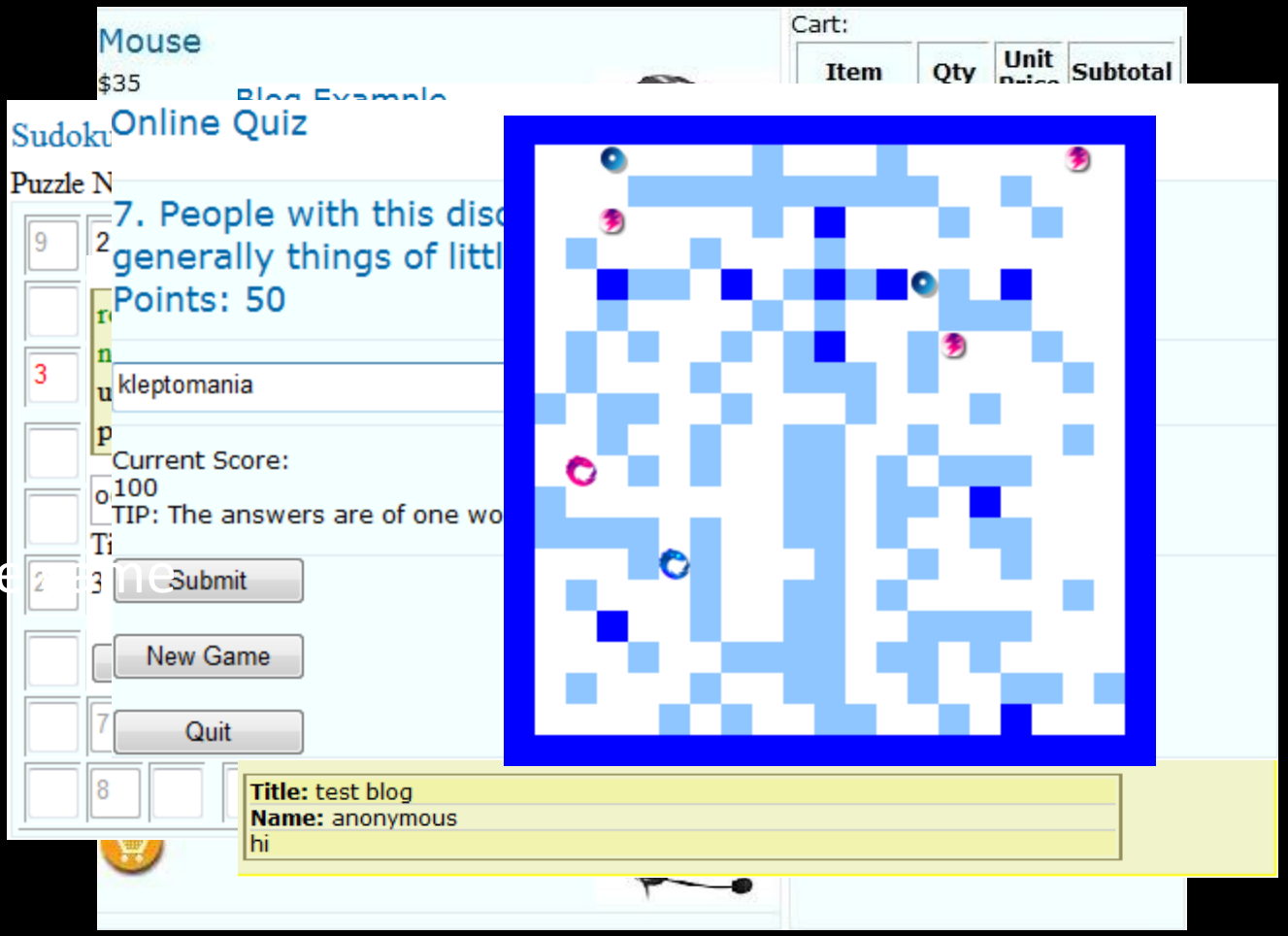


1. Keep a replica of the client code
2. Capture user events & transmit to server for replay
3. Compare server and client results

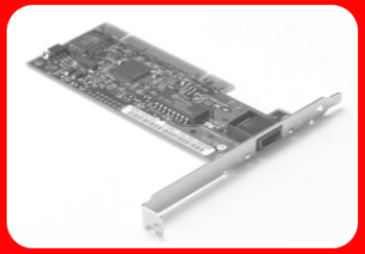
Experiments

Ripley Applications

- ✓ Shopping cart
- ✓ Sudoku
- ✓ Blog
- ✓ Speed typing
- ✓ Online Quiz
- ✓ Distributed online

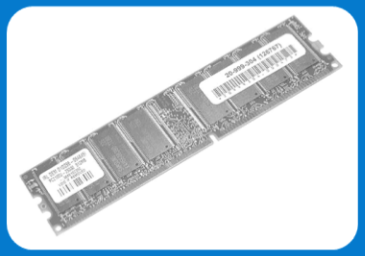


Performance Overhead: Volta Benchmarks



Network:

- 2-3 bytes per user event (key press, mouse, etc.)
- Event stream compresses extremely well



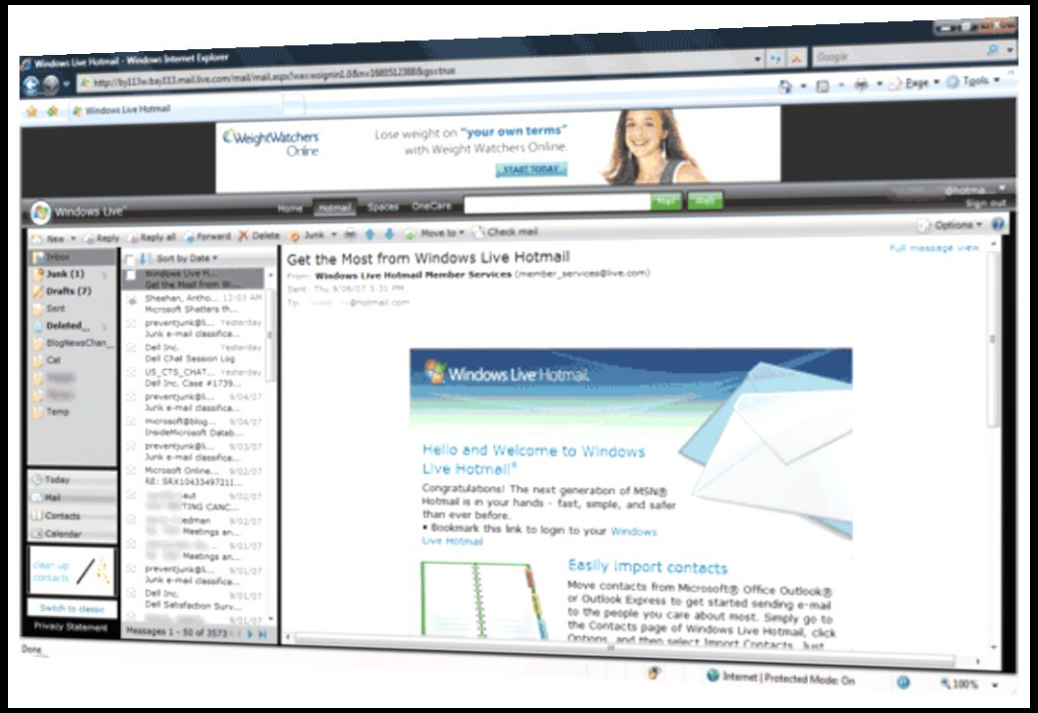
Memory:

- About 1 MB per connected client
- Can scale to 1,000's of clients per server

CPU:

- Client: Several *ms* of overhead added for event capture
- Server: Several *ms* for server-side checking

Replicating Hotmail



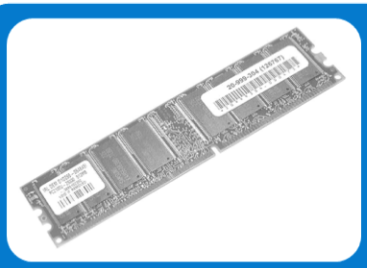
- Hotmail size
  - 793 KB download
  - 703 KB JavaScript
  - 31,000+ lines of code

- 10 minutes of normal use
- Requests: 617 KB
- Responses: 3,045 KB

Replicating Hotmail

Ripley traffic:

- 491 keyboard & mouse events
- 1.4% without compression (8.6 KB)
- 0.4% otherwise (2.8 KB)



Memory:

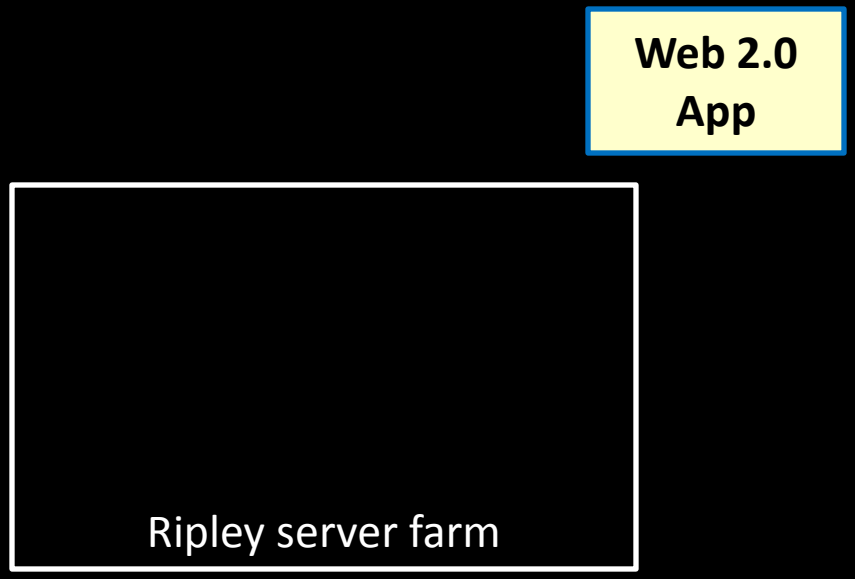
- DOM state in memory: 350 -- 450 KB
- JavaScript heap state: 1.3 MB
- < 1.75 MB in total
- Can scale up to hundreds of clients

CPU overhead small:

- Most: < 15 *ms*
- Email message processing: 125 *ms*
- Most time spent in HTML rendering and data marshaling code

Ripley: Vision for the Future

- Secure-by-construction Software + Services

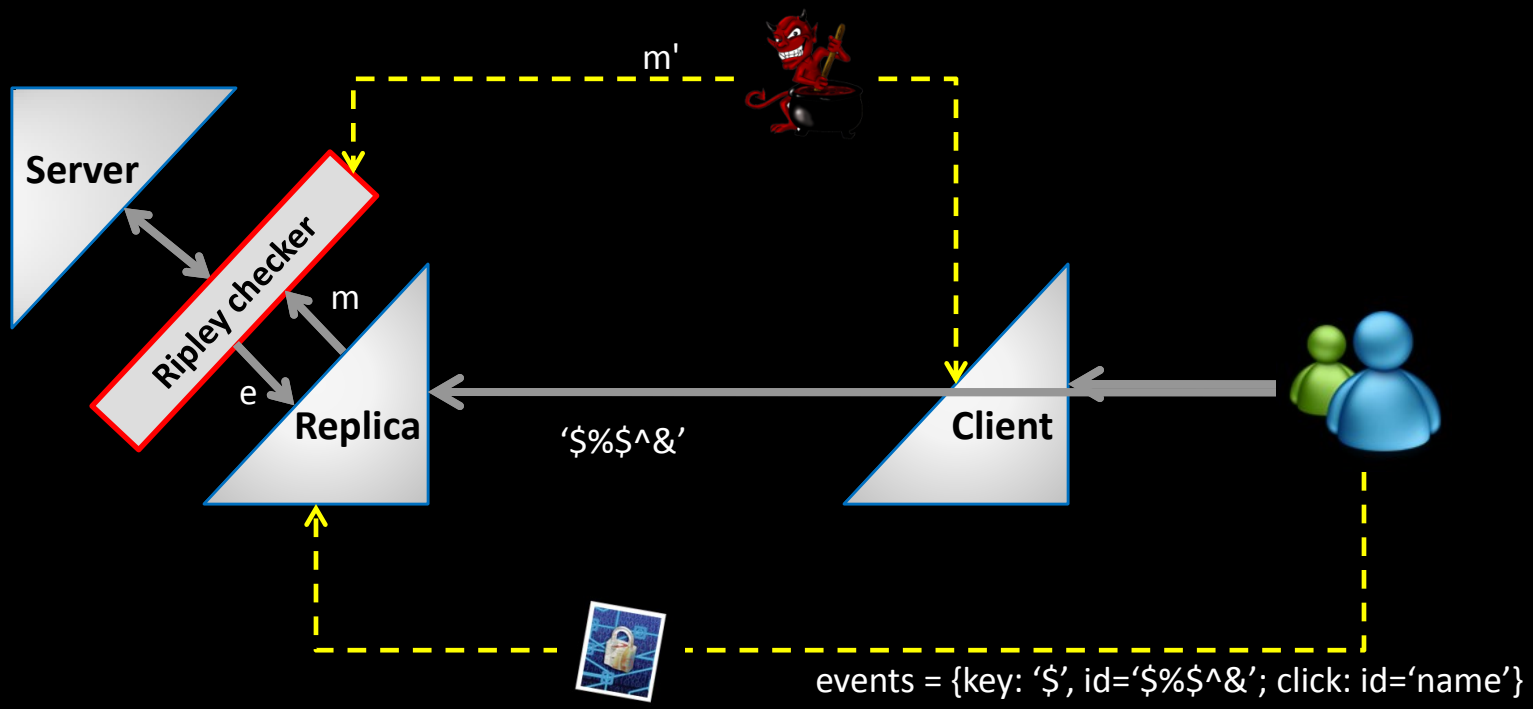


Contact us

Ben Livshits (livshits@microsoft.com)

Microsoft Research Ripley project

Malicious Event Stream



Every attack against integrity of the Ripley-protected application was possible against the standalone app