RTC-Web Security Model Overview

W3C TPAC 2011

Eric Rescorla

ekr@rtfm.com

Context

- IETF RTC-Web has been developing a security model
- This is an attempt to summarize the current state

The Browser Threat Model

Core Web Security Guarantee: "users can safely visit arbitrary web sites and execute scripts provided by those sites." [HCB$^+$11]

- This includes sites which are hosting malicious scripts!
- Basic Web security technique is isolation/sandboxing
  - Protect your computer from malicious scripts
  - Protect content from site A from content hosted at site B
  - Protect site A from content hosted at site B
- In this case we're primarily concerned with JavaScript running in the browser

The browser acts as a trusted computing base for the site

The Internet Threat Model

“You hand the packets to the attacker to deliver.” – Steve Bellovin

- Endpoints are secure
- An attack has complete control of the network...
  - Can read any packet you send
  - Can modify any packet in transit
  - Can forge a packet with any source address
- Cryptographic techniques are the major security measure

So what's our threat model?

- Solid protection under the browser threat model
  - Safe to visit any Web site
  - Can safely authorize A and not B
- Do the best we can under the Internet threat model
  - Protect against network attackers if people use HTTPS
  - Provide strong media security for RTCWeb-RTCWeb calls

List of Issues to Consider

- Access to local devices
- Consent to communications
- Communications security

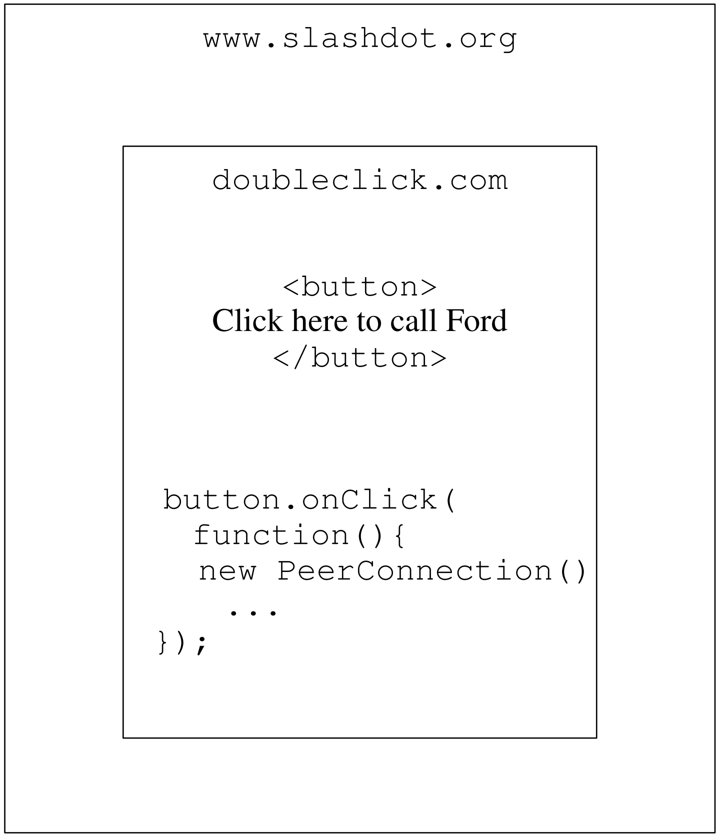
Access to Local Devices

- Making phone (and video) calls requires that your voice be transmitted to other side
  - But the *other side* is controlled by some site you visit
  - What if you visit `http://bugmyphone.example.com`?
- Somehow we need to get the user's consent
  - But to what?
  - And when?
  - Users routinely click through warning dialogs when presenting "in-flow"

Permissions Models

- Short-term permissions
  - Allow this single call
  - E.g., for a customer service call
- Long-term permissions
  - Allow this site to make calls whenever it wants
  - E.g., for a social networking or calling site
  - Is this going to happen? [Terriberry]
- Per-peer permissions
  - Allow calls to `bob@example.com`
  - Requires strong peer identity

Ad Hoc Calling from Embedded Advertisements



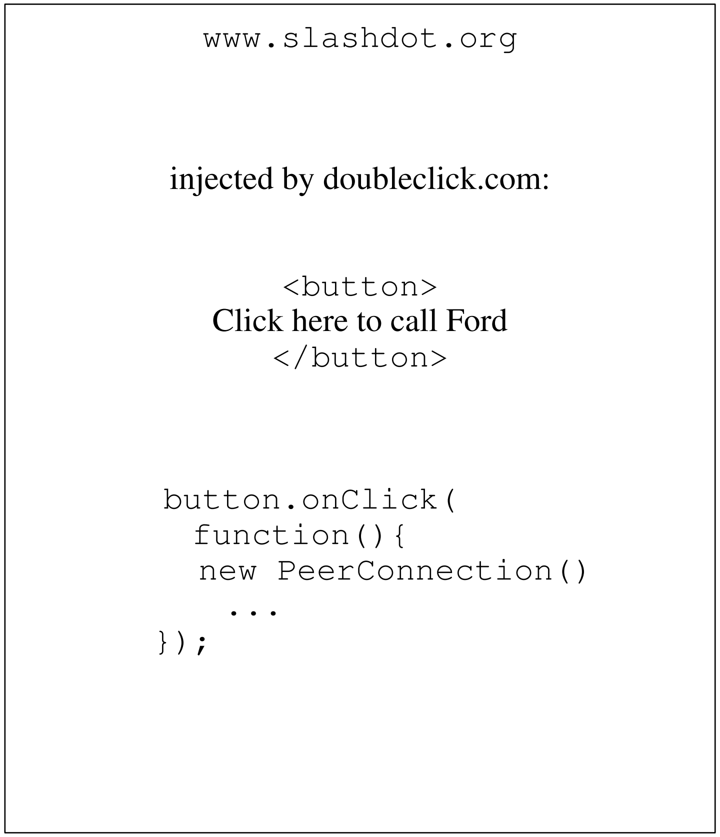
www.slashdot.org

doubleclick.com

```
<button>
Click here to call Ford
</button>
```

```
 button.onClick(
  function(){
   new PeerConnection()
     ...
});
```

Option A: Ad in an IFRAME



www.slashdot.org

injected by doubleclick.com:

```
<button>
Click here to call Ford
</button>
```

```
 button.onClick(
  function(){
   new PeerConnection()
     ...
});
```

Option B: Injected ad

User expectations

- When I place this call I'm talking to Ford
- Not giving Ford long-term access to my camera and microphone

- But I'm on Slashdot...
  - Do I think Slashdot has endorsed this?

UI for Short-Term Permissions

- Generally necessary at the time of call
- Needs an immediate call-to-action
  - E.g., a “check-my-hair” self video image with an OK button
  - This must be in the browser chrome to prevent click-jacking
- Permissions should only be good for this call
- Non-maskable indicator of call status in browser chrome
- Also a non-maskable call termination button

API impact of short-term permissions

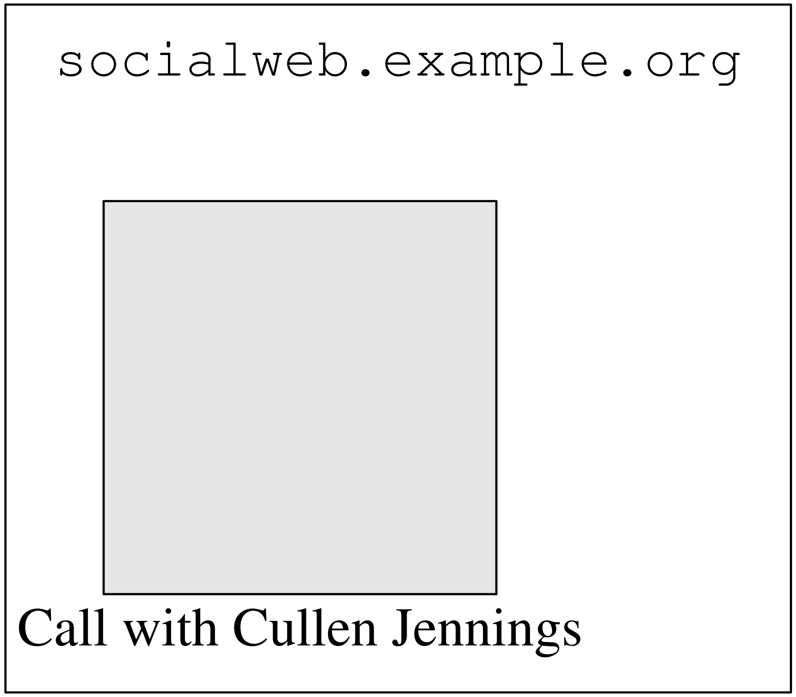
- One natural design is to show “self” picture a la Facetime
  - Here’s your video, do you want to set up the call
- But this implies *some level* of device access prior to permissions grant
  - Step 1: display video to user with call start button
  - Step 2: start call
- Out of scope?

What about the site I'm visiting?

- Adam Barth: the user thinks he's on Slashdot
  - Even though Slashdot neither placed the ad nor is the called party
  - Only vaguely conscious of ad networks
- Should the top-level site get to have an opinion?
  - Protect the user?
  - Protect its reputation?
  - What about privacy?

Long-term permissions: Calling Services

- I have an account on SocialWeb
  - ... “friends” with a bunch of my real-world friends
  - Want to call one of my friends



socialweb.example.org

Call with Cullen Jennings

UI for Long-Term Permissions

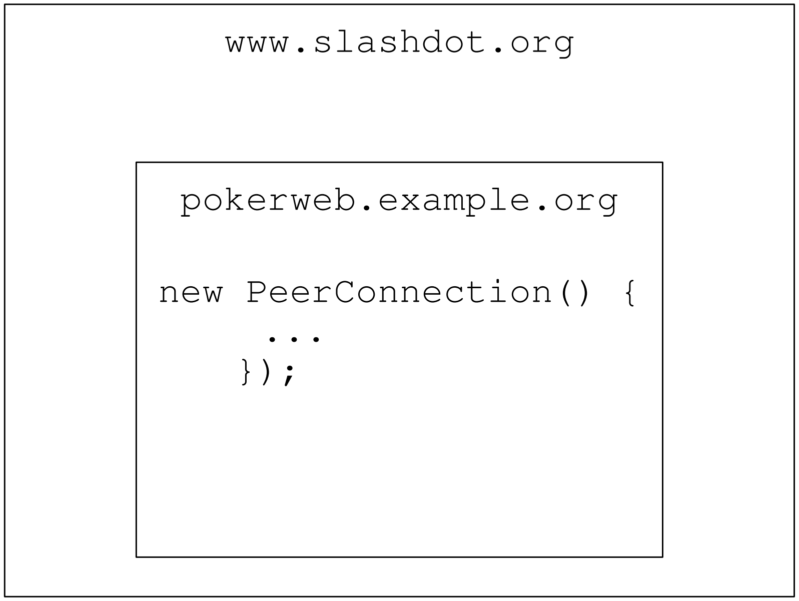
- Probably a bad idea to have an intrusive dialog
  - Want to avoid “click here to screw yourself”
  - Possible: door hanger style UI “some features on this site may not work”
- Do we want this at all?
  - Tension between convenience and security
    - ∗ Calling sites likely want a clean UI
    - ∗ But this means that the calling site has a lot of power
  - Is this a permissions model the browsers want?

Peer Identity-based Permissions

- Not clear what level of attention to demand
- General idea
  - Do you want to talk to `bob@example.com`
  - Easier if you can integrate with address book
- Orthogonal to site-based permissions?

Partial Digression: Network Attackers

- Assumption: I've authorized PokerWeb
- I'm in an Internet Cafe and visit any URL
  - Attacker injects IFRAME pretending to be PokerWeb
  - But calls go to him



www.slashdot.org

pokerweb.example.org

```
new PeerConnection() {
      ...
    });
```

- Result: attacker has bugged your computer

User Expectations

- It's safe to authorize PokerWeb and then surf the Internet
  - Without being bugged
- Including on insecure networks
  - This may include your home network
- Unfortunately, this is not true here...

Origin and HTTP/HTTPS

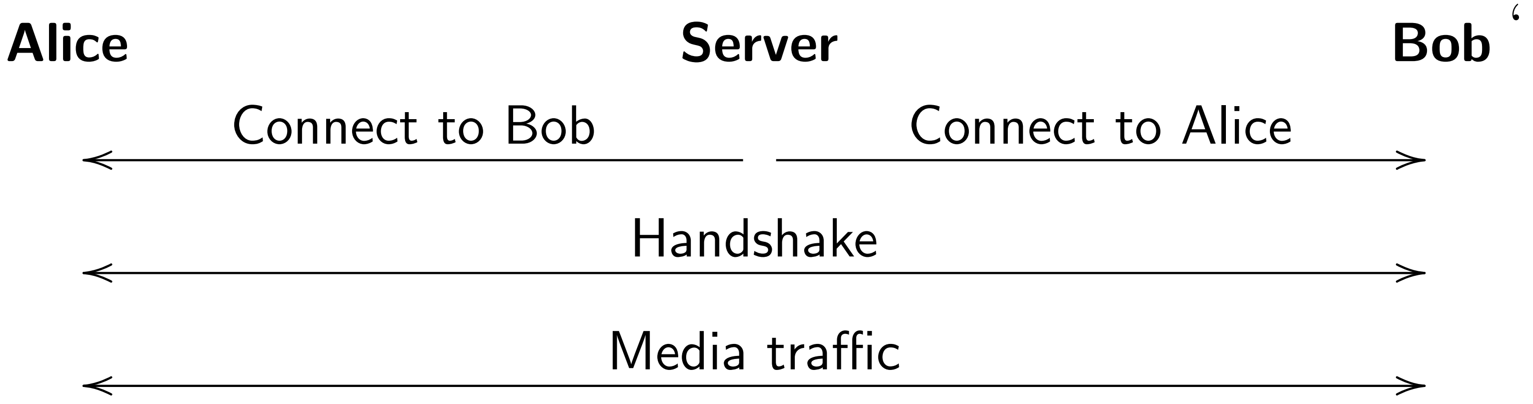
- Basic problem here is HTTP-based origins
  - Question: should RTC-Web be available over HTTP
- What about mixed content?
  - Appears to be HTTPS but for our purposes is HTTP
  - Treat as HTTP?
  - Forbid RTC-Web entirely over mixed content?

Consent for real-time peer-to-peer communication

- Need to able to send data between two browsers
  - Unless you want to relay everything
- But this is unsafe (and violates SOP)
  - Not OK to let browsers send TCP and UDP to arbitrary locations
- General principle: verify consent
  - Before sending traffic from a script to recipient, verify recipient wants to receive it from the sender
  - Familiar paradigm from CORS [vK10] and WebSockets[Fet11]

How to verify communications consent for RTC-Web

- Can't trust the server (see above)
  - Needs to be enforced by the browser
- Browser does a handshake with target peer to verify connectivity



Alice **Server** **Bob**

Connect to Bob

Connect to Alice

Handshake

Media traffic

- This should look familiar from ICE [Ros10]

Implementing Communications Consent Securely

- Remember: we don't trust the JS

- Restrict pre-handshake communications
  - Restrict communications to an endpoint until handshake completes
  - Minimize application control of ICE packets (extensions, etc.)
  - Rate-limit ICE checks
- Browser **must not** let application see STUN transaction ID
  - Prevents forgery of STUN responses by the server
- What about cross-protocol attacks?
  - Not really an issue for UDP, especially with DTLS
  - TCP **must** use masking

What about communications security?

- We've already addressed this in the context of SIP
  - Things aren't that different here–all the usual protocols work
- This should be mostly invisible with current style API
  - The relevant messaging is embedded in the SDP
  - Possibly need API controls to set up security parameters
  - ... cipher suites, etc.
- Security indicators need to be in the browser chrome
  - So the it can't be changed by the JS

Questions?

References

[Fet11] Ian Fette. The WebSocket protocol. `draft-ietf-hybi-thewebsocketprotocol-06.txt`, February 2011.

[HCB$^+$11] Lin-Shung Huang, Eric Y. Chen, Adam Barth, Eric Rescorla, and Collin Jackson. Talking to Yourself for Fun and Profit. In *W2SP 2011*, 2011.

[Ros10] J. Rosenberg. Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols. RFC 5245, 2010.

[vK10] Anne van Kesteren. Cross-Origin Resource Sharing. `http://www.w3.org/TR/access-control/`, 2010.