



# Designing Secure AIR Applications



- **Plan for Success**
- **Consider security before you have lots of users**
  - **Updates - Fix security issues**
  - **Injection - Prevent attackers from controlling your application**

# Updates: Stuff Happens



- Sooner or later you will need to issue an update
  - A security issue could make this urgent
    - ▶ Updates require advance planning
- Make updates work and *then* implement features

## Updates: Division of Responsibility



- Applications implement update *policies*
  - When to download
  - How to notify
  - Version numbering scheme
    - ▶ Updater frameworks are available
- AIR provides an update *capability*
  - Handles install, elevation, runtime updates

## Updates: Downgrade Attacks



- Your update mechanism is a point of attack
- Successful attack installs an old version
  - Older version contains a vulnerability
  - Signing doesn't protect against this
  - Attacker can just distribute old .air files

# Updates: Defeating Downgrade Attacks



- You *must* check version numbers during updates
  - Why the Updater API requires a version #
- You *should* declare `<customUpdateUI>`
  - Gives complete control over installing updates

# Import and Injection: Understanding Loading



- Executable content (SWF, HTML/JS) can be loaded different ways
- *Sandboxed*
  - Stays in its own domain and cannot reach out
- *Imported*
  - Runs in loader's sandbox with loader's privilege

# Import and Injection: Examples



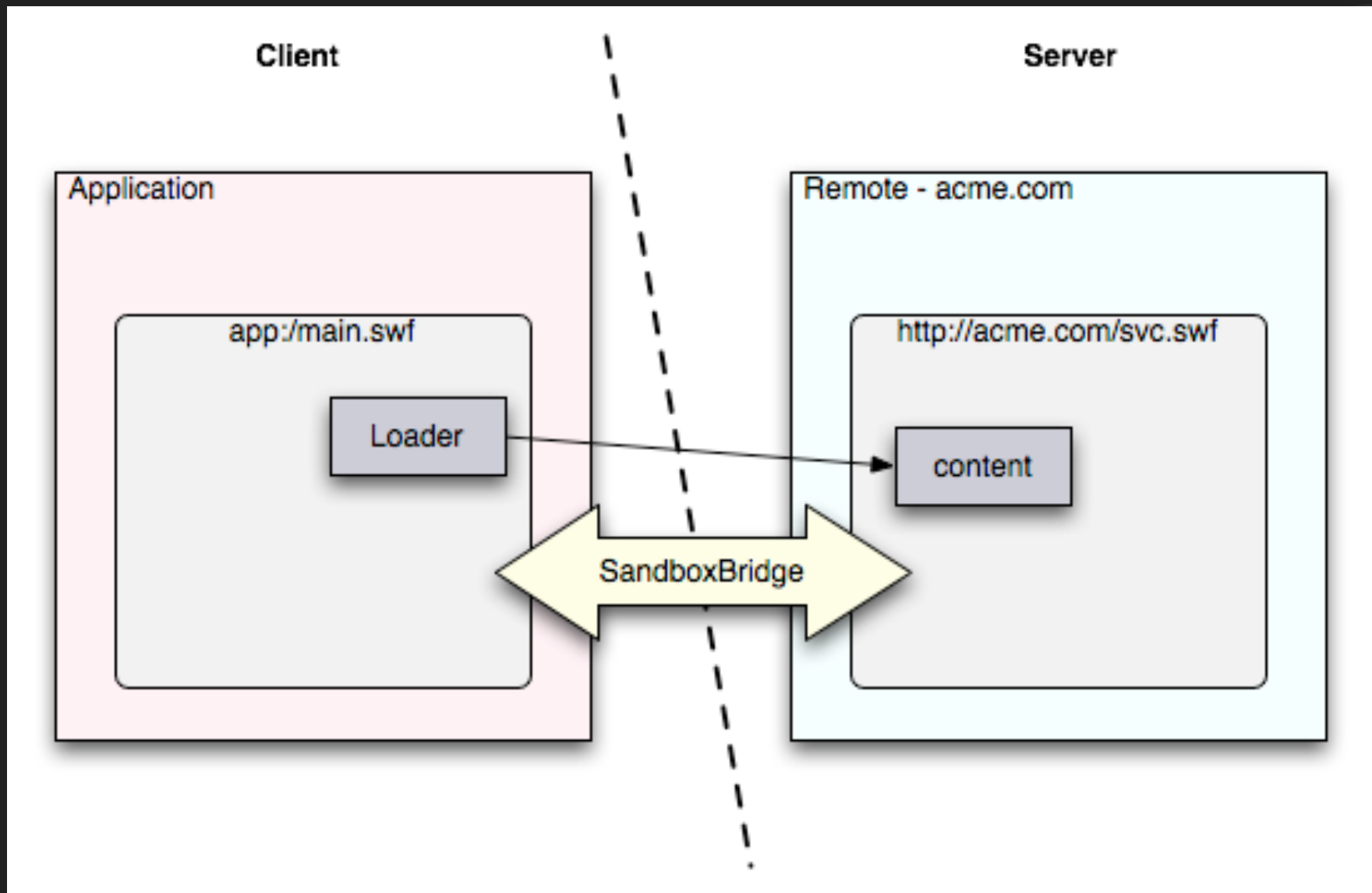
- *Sandboxed*

- HTML `<FRAME>`, `<IFRAME>`, `<IMG>` tags
- SWF `Loader.load()`, Flex `SWFLoader`

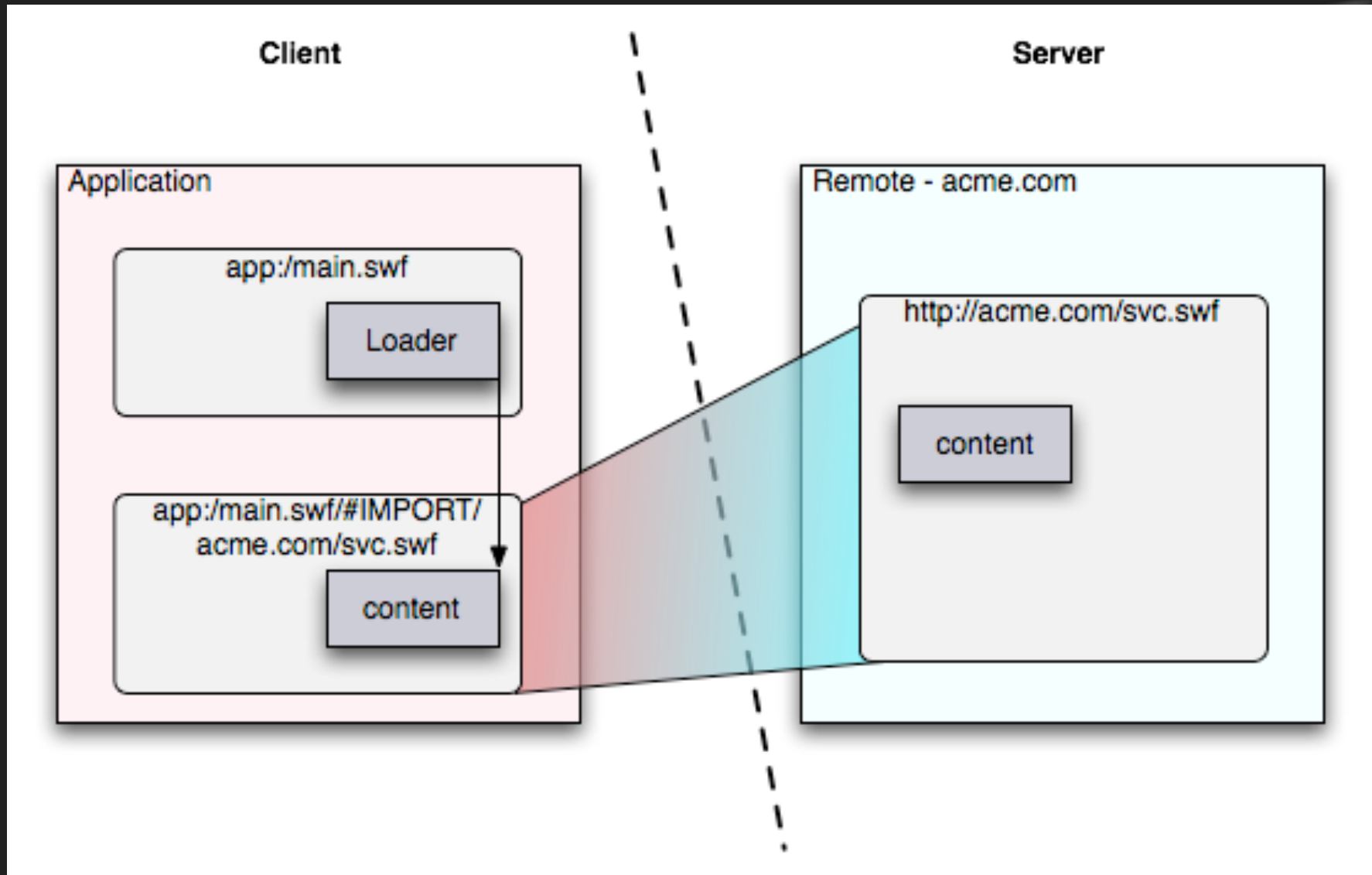
- *Imported*

- JS `eval()`, `innerHTML`, `new Function("...")`
- HTML `<SCRIPT>` tag
- SWF `Loader.loadBytes()`

# Import and Injection: Sandboxed



# Import and Injection: Imported



## Import and Injection: Attacks



- Importing untrustworthy content opens you to *injection attacks*
- Injected content has full privilege
  - In browser: access to server
  - In AIR: access to client machines!
- Attacks enabled by common coding patterns, especially with AJAX, cross-domain JSON
  - ▶ AIR application sandbox provides some protection

## Import and Injection: Keep Your Distance



- Often better to load content in its own sandbox
- AIR adds communication across sandboxes - *SandboxBridge*
  - *Pass-by-value*: no references leaked
  - Opt-in on both sides
  - Good for untrusted plugins
- Don't abuse SandboxBridges by exposing dangerous APIs

# Import and Injection: Trust Imported Content



- Import can be valuable
  - Modules; reducing download size
- Only import content you can *prove you trust*
- How to prove it?
  - Server name? *No*. DNS attacks win
  - Server + SSL/TLS? Better, but *No*. Server vulnerabilities put clients at risk
  - Signed content? *YES!* Proof it came from you

Dziękuję!



- <http://weblogs.macromedia.com/ema1asky>