adding origins to our model

HTTPEvent

from, to, origin

causes

EndPoint

Request

Response

Embedded Request

Redirect Response

Client

Server

response

embeds
analysis steps
analysis steps

› define origin tracking & policy
› express security property
› check that policy implies property
define origin tracking

abstract sig EndPoint { causes: set HTTPEvent }
sig Client, Server extends EndPoint {}
abstract sig HTTPEvent { from, to: EndPoint }
sig Request extends HTTPEvent { response: Response }
sig Response extends HTTPEvent { embeds: set Request }
sig Embedded extends Request {}
sig Redirect extends Response {}

fact Origin {
    // for a redirect, origin is same as request, else server
    all r: Request | r.response.origin =
        (r.response in Redirect implies r.origin else r.response.from)

    // embedded requests have the same origin as the response
    all r: Response, e: r.embeds | e.origin = r.origin

    // requests that are not embedded come from the client
    all r: Request - Embedded | r.origin = r.from
}
define policy

```plaintext
pred appliesSOP (s: Server) {
    // request is only accepted if origin is server itself or sender
    all r: Request | r.to = s implies r.origin = r.to or r.origin = r.from
}
```
check {
  no server: Server, attacker: Server - server {
    // no direct request to attacker
    no r: Request | r.to = attacker and r.origin in Client
    // trusted server obeys origin policy
    server.appliesSOP
    // and attacker still gets request through
    some r: attacker.causes | r.to = server
  }
} for 6 but 1 Client, 2 Server
attack!
attack!
attack!
attack!
attack!
attack!