University of Stuttgart
Institute of
Information Security

WIM: An Expressive Formal Model of the Web Infrastructure

Ralf Küsters

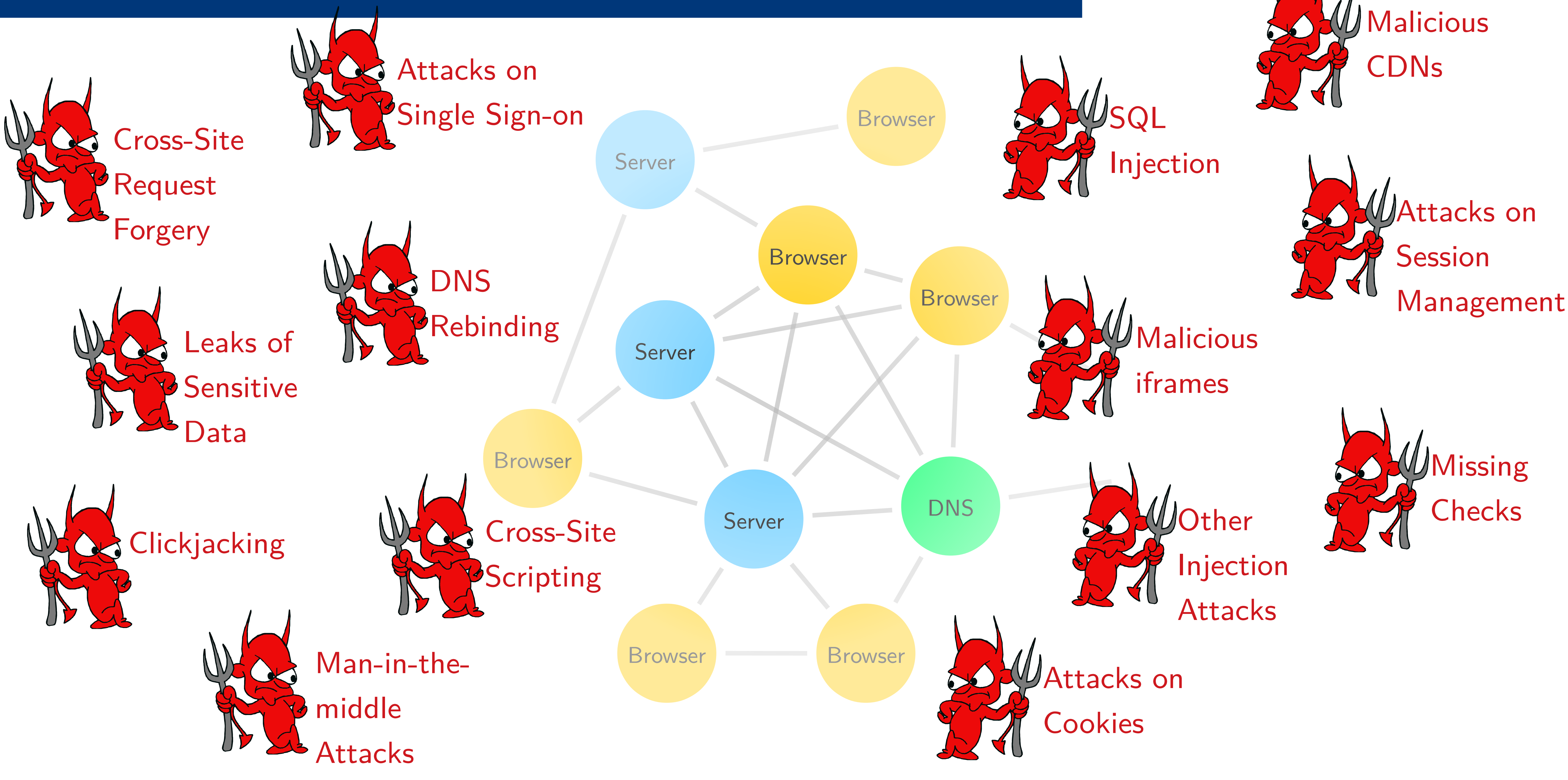
2019-06-27

Joint work with

Daniel Fett, Pedram Hosseyni, and Guido Schmitz

[S&P 2014, ESORICS 2015, CCS 2015, CCS 2016, CSF 2017, S&P 2019]

Many Web Attacks...



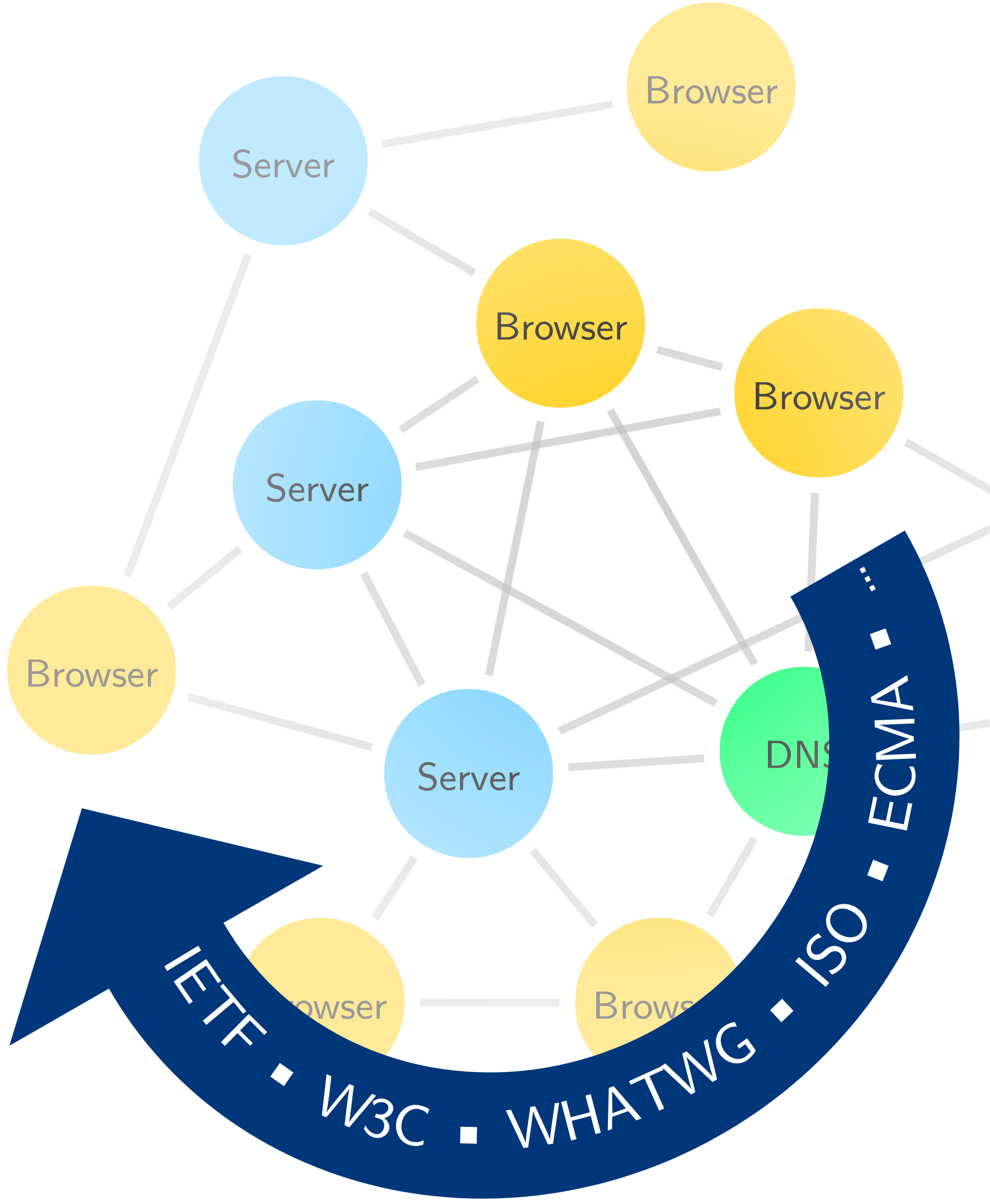
...but why?

The web is complex ...

- Network of heterogeneous components
- Large number of complex standards developed at a high pace by many separate organizations

... and web applications as well.

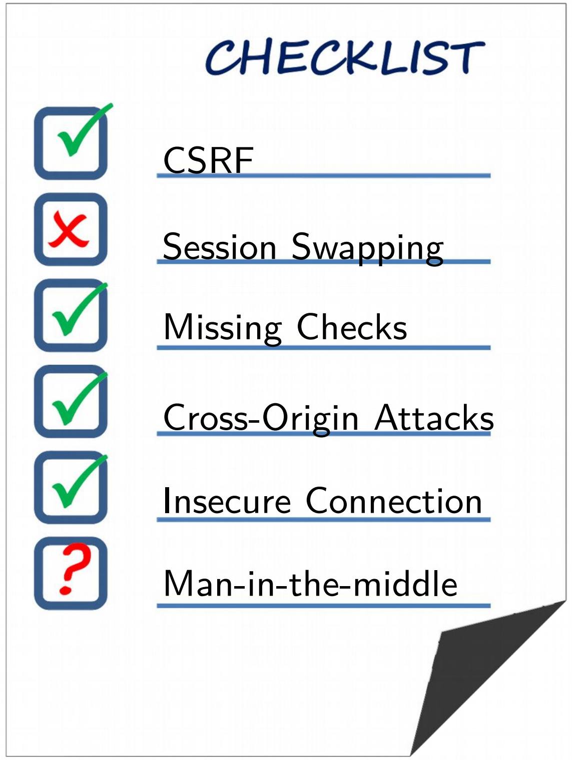
- More features, more interaction
- Many bugs and errors

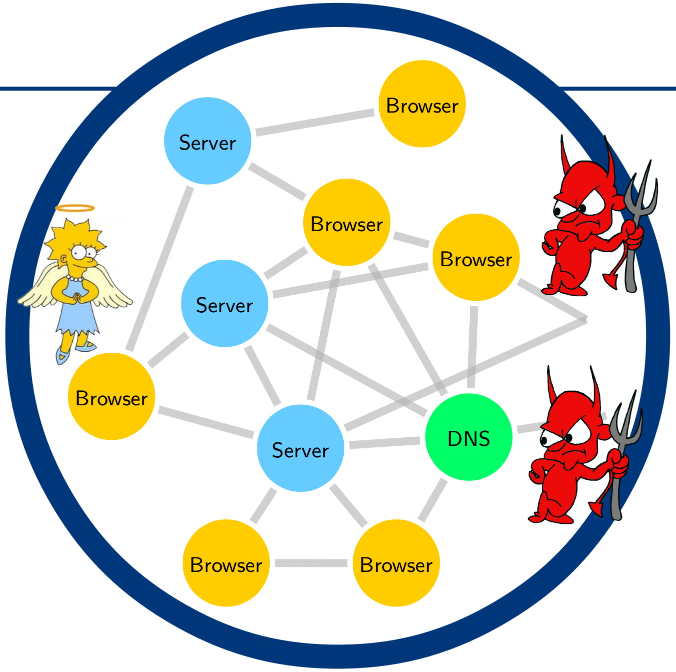


Finding Vulnerabilities: Current Practice

Expert review

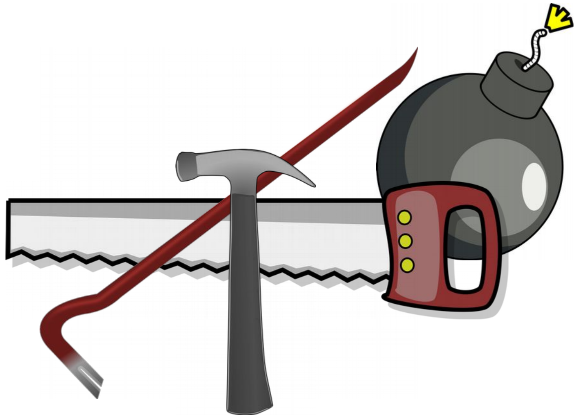
of standards and implementations





Penetration testing

using tools or manual analysis

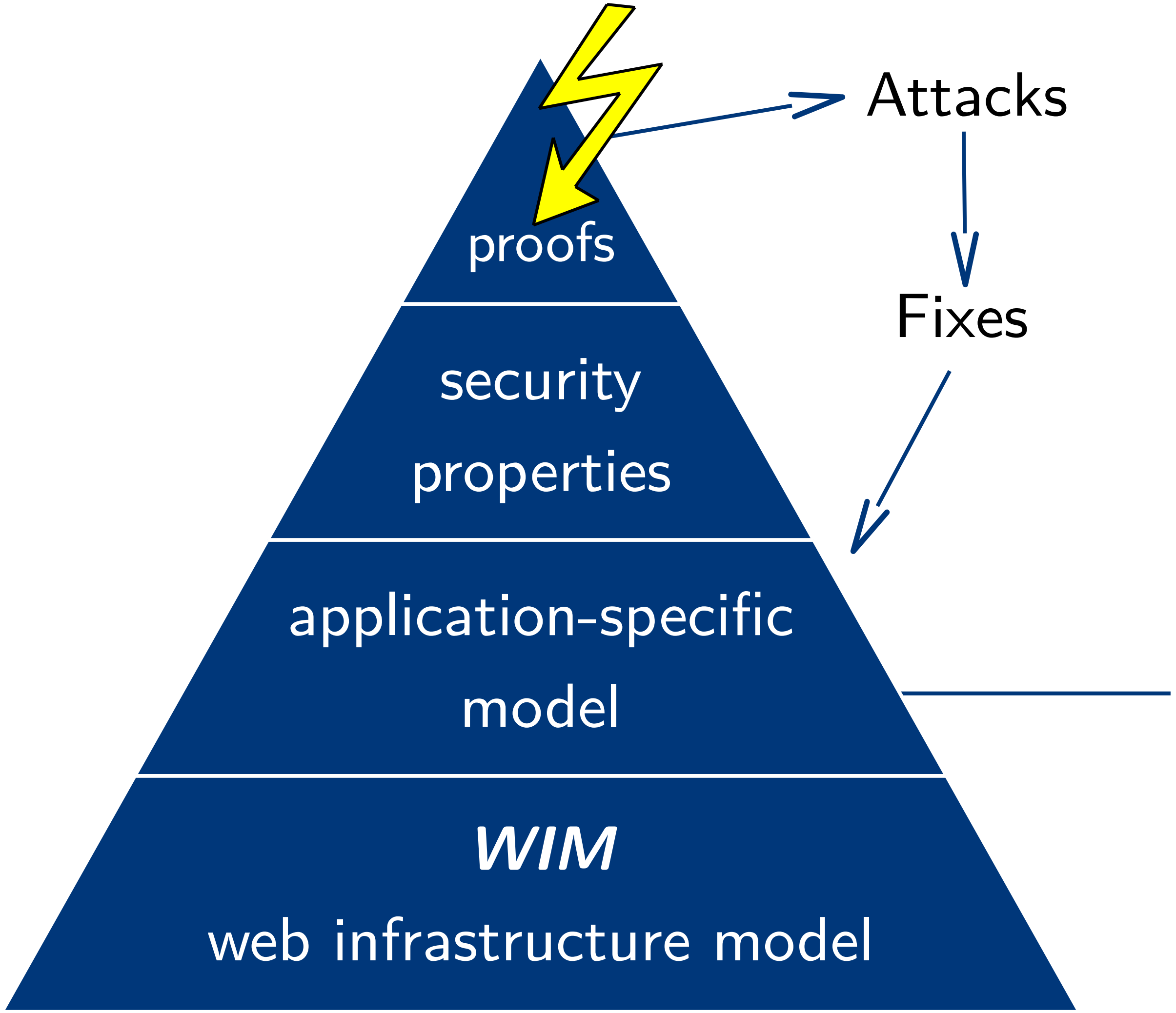


Downsides

- It is easy to miss attacks, even for experts
- Pentesting focuses on known attacks
- Finding new attack types depends on the creativity of the experts
- Both methods do not guarantee security, not even for a limited set of attacks

➡ **Can we develop a more systematic way of finding vulnerabilities, and even prove security (in a meaningful model of the web infrastructure as a whole)?**

Our Model-Based Approach

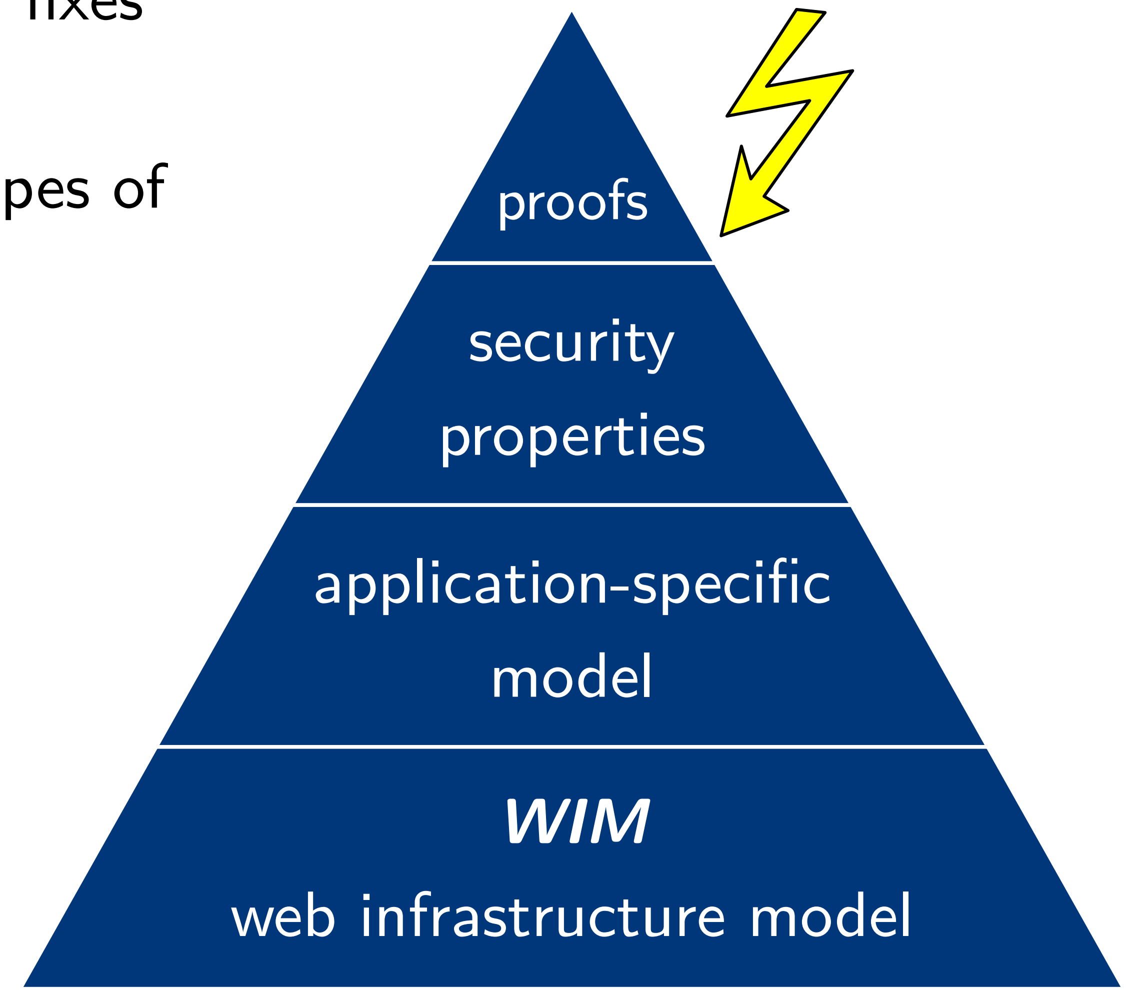


For instance:
Single Sign-On Standards and Applications (OAuth, OIDC, Financial-grade API, etc.)

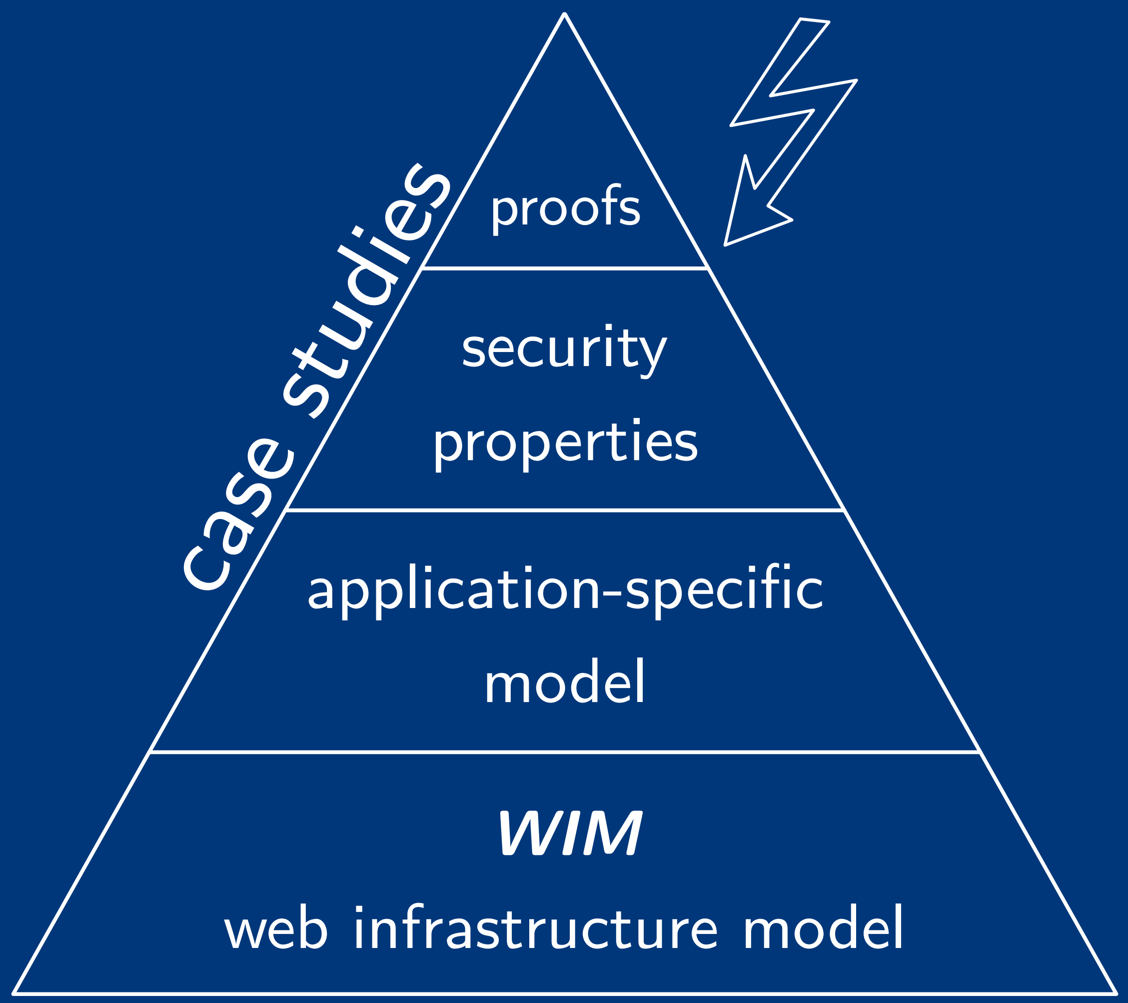
Advantages

This approach can yield...

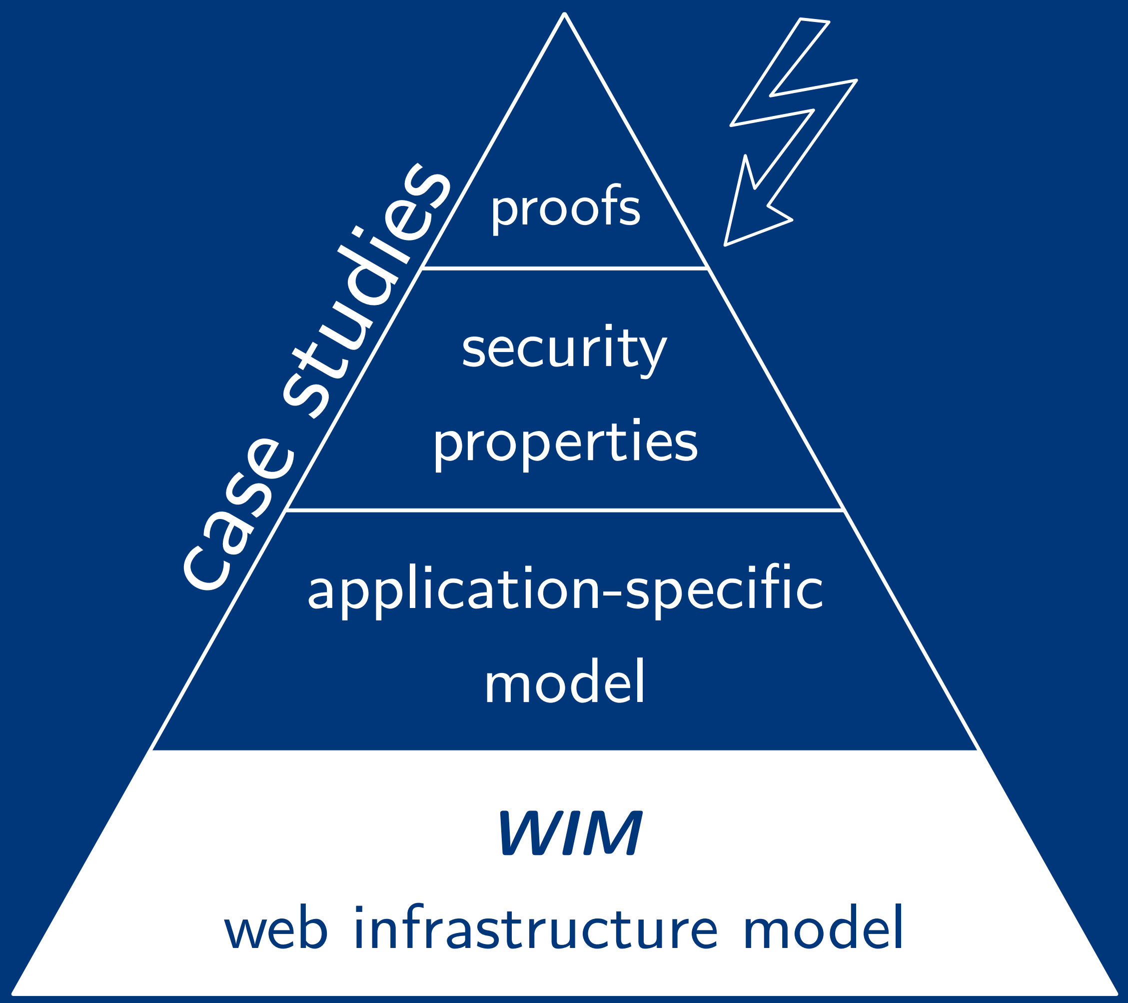
- new attacks and respective fixes
- strong security guarantees excluding even unknown types of attacks



An Expressive Formal Model of the Web Infrastructure

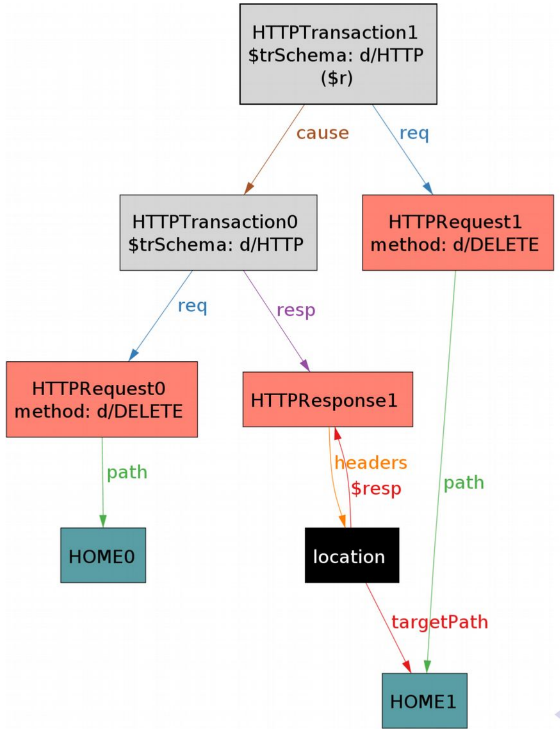


An Expressive Formal Model of the Web Infrastructure



Prior Web Models

- [Kerschbaum 2007]
  Analysis of CSRF protection in the Alloy model checker
- [Akhawe, Barth, Lam, Mitchell, Song 2010]
  First formal "web model", in Alloy, five case studies
- [Bansal, Bhargavan, Maffeis et al. 2012, 2013, 2014]
  Formal web model with many web features, based on ProVerif tool,
  new attacks on encrypted cloud storage and OAuth 2.0



Very limited web models
Limitations and constraints of tools (e.g., encoding of messages/terms and data structures)

Our approach: goal was a very detailed, close-to-standards web model, (started with) pen-and-paper.

Further Related Work (Formal Analysis)

- [Kumar et al., 2011-2014]: Alloy-based with BAN logic
- [Bai et al., 2013]: AuthScan + ProVerif
- [Bohannon and Pierce, 2010]
  - "Featherweight Firefox"
  - Information Flow tracking in web browser core
  - No security policies by default
- [Sabelfeld et al. 2016]: Information-flow security for JavaScript and its APIs
- [Börger et al., 2012]
  - Abstract State Machines
  - Focus on web server, limited browser model

The Web Infrastructure Model ***WIM***

- Detailed, comprehensive, and precise formal model

  Network interactions

  Attacker behavior

  DNS servers

  Generic web server model

  Web browsers

- Summarizes and condenses relevant standards

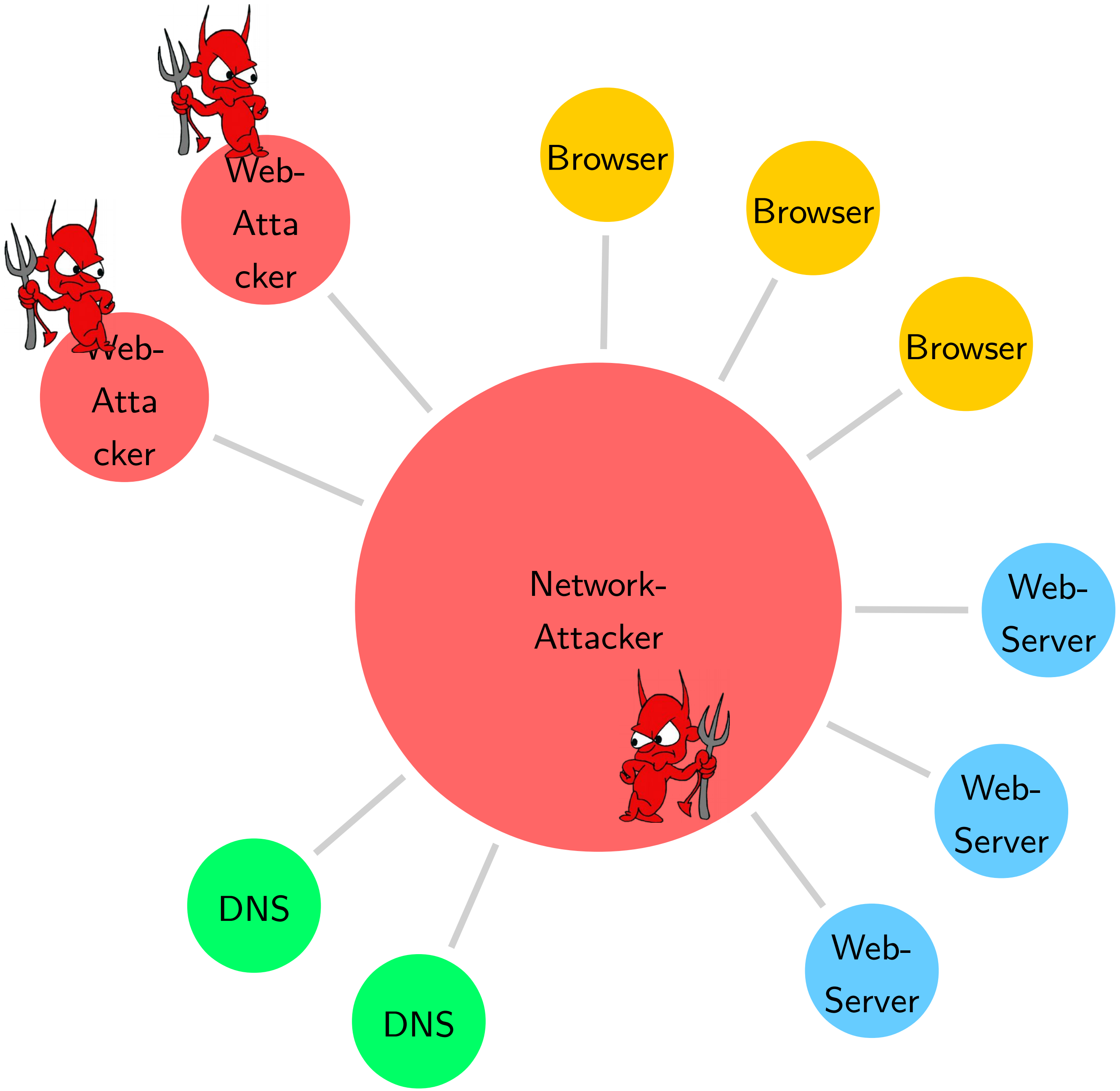
- Solid basis for security and privacy analyses

  of web standards and applications

- Reference model

  developers, researchers, teaching, and tool-based analysis

WIM Network Model and Attackers



Dolev-Yao-Style Model:

- Messages are terms
- Attacker, Browsers, Servers, Scripts (honest or malicious) are Dolev-Yao processes
- **Not** just a standard Dolev-Yao model for protocol analysis, but rather covers web features, close to web standards.

The Web Infrastructure Model ***WIM***

- Detailed, comprehensive, and precise formal model

  Network interactions

  Attacker behaviour

  DNS servers

  Generic web server model

  Web browsers

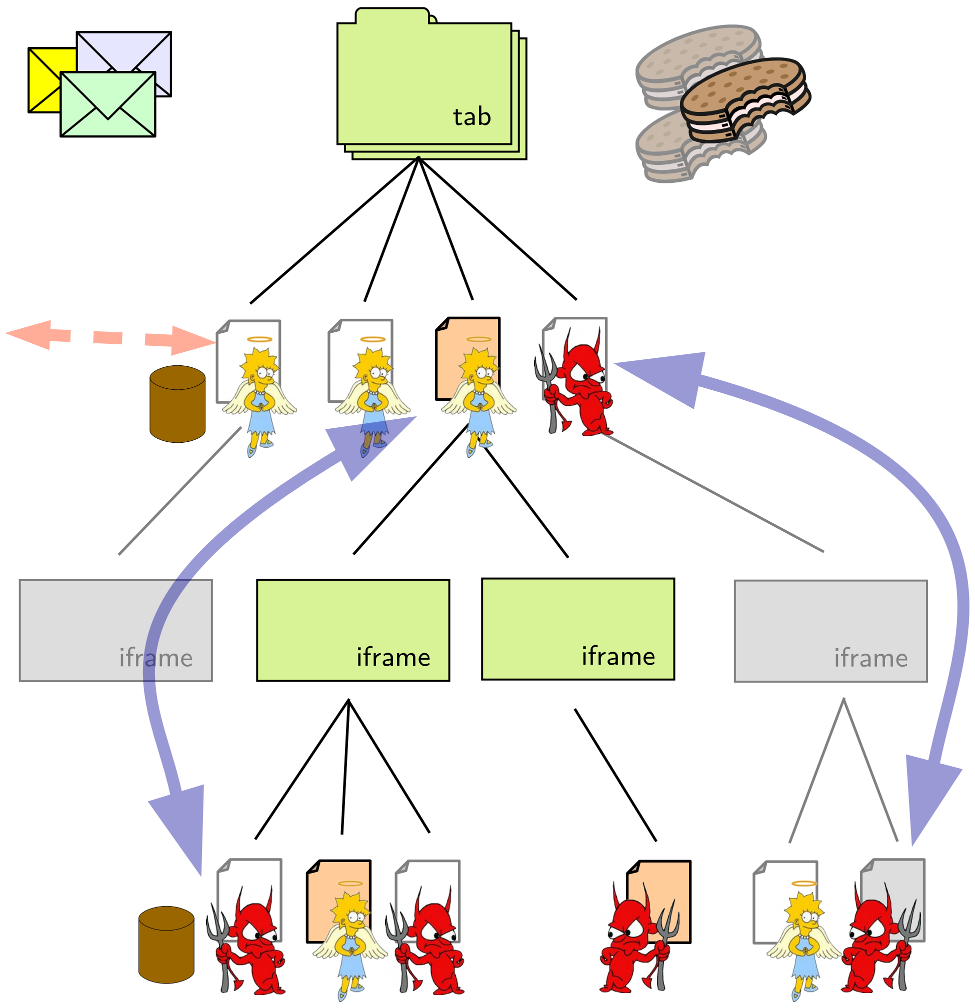
- Summarizes and condenses relevant standards
- Solid basis for security and privacy analyses

  of web standards and applications

- Reference model

  developers, researchers, teaching, and tool-based analysis

WIM Web Browser Model



Including …

- DNS, HTTP, HTTPS
- window & document structure
- scripts
  (honest and malicious)
- web storage & cookies
- web messaging & XHR
- message headers
  (Origin, STS, Location, Referer, …)

Origin: https://example.com

- redirections
- security policies
- WebRTC
- dynamic corruption
- …

WIM Web Browser Model - Example

quite complex rules

Algorithm 8 Web Browser Model: Process an HTTP response.

```
function PROCESSRESPONSE(response, reference, request, requestUrl, key, f, s′)
    if Set-Cookie ∈ response.headers then
        for each c ∈⟨⟩ response.headers[Set-Cookie], c ∈ Cookies do
            let s′.cookies[request.host]
             ↪ := AddCookie(s′.cookies[request.host], c)
    if Strict-Transport-Security ∈ response.headers ∧ requestUrl.protocol ≡ S then
        let s′.sts := s′.sts +⟨⟩ request.host
    if Referer ∈ request.headers then
        let referrer := request.headers[Referer]
    else
        let referrer := ⊥
    if Location ∈ response.headers ∧ response.status ∈ {303, 307} then
        let url := response.headers[Location]
        if url.fragment ≡ ⊥ then
            let url.fragment := requestUrl.fragment
        let method′ := request.method
        let body′ := request.body
        if Origin ∈ request.headers then
            let origin := ⟨request.headers[Origin], ⟨request.host, url.protocol⟩⟩
        else
            let origin := ⊥
        if response.status ≡ 303 ∧ request.method ∉ {GET, HEAD} then
            let method′ := GET
            let body′ := ⟨⟩
```

The Web Infrastructure Model *WIM*

- Detailed, comprehensive, and precise formal model

  Network interactions

  Attacker behaviour

  DNS servers

  Generic web server model

  Web browsers

- Summarizes and condenses relevant standards
- Solid basis for security and privacy analyses

  of web standards and applications

- Reference model

  developers, researchers, teaching, and tool-based analysis

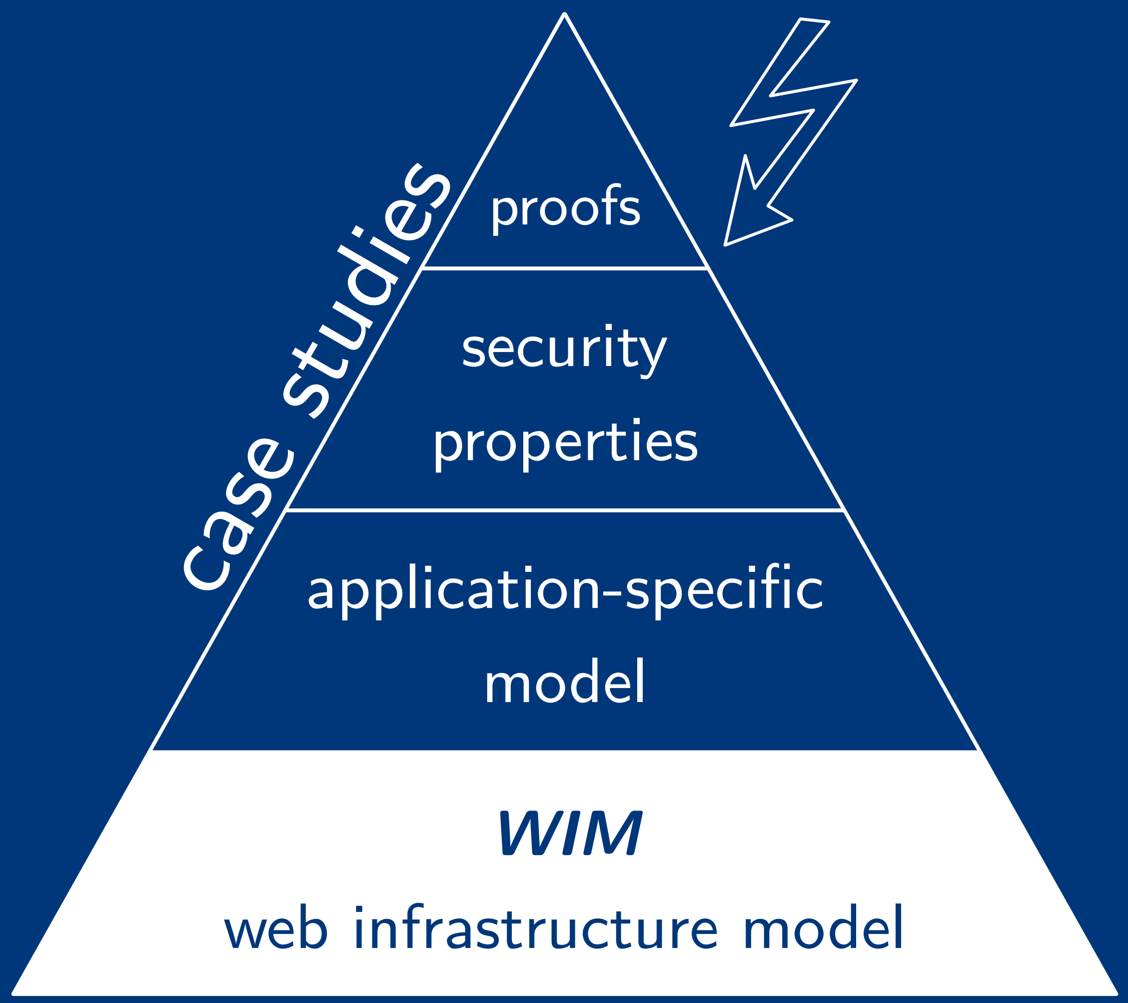
Limitations

- No language details
- No user interface details (e.g., no clickjacking attacks)
- No byte-level attacks (e.g., buffer overflows)
- Abstract view on cryptography and TLS

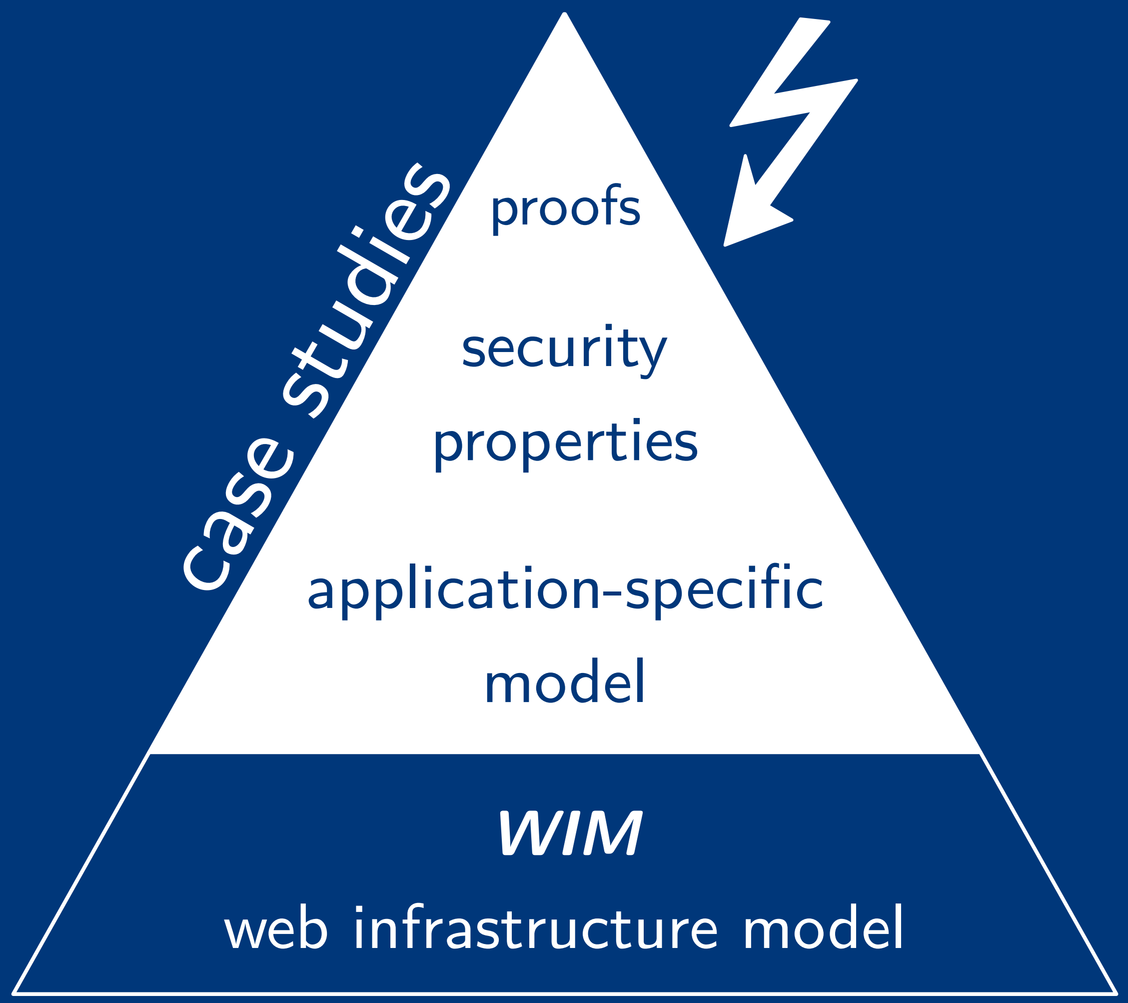
Model can in principle be extended to capture these aspects as well.

Trade-off: comprehensiveness vs. simplicity

An Expressive Formal Model of the Web Infrastructure

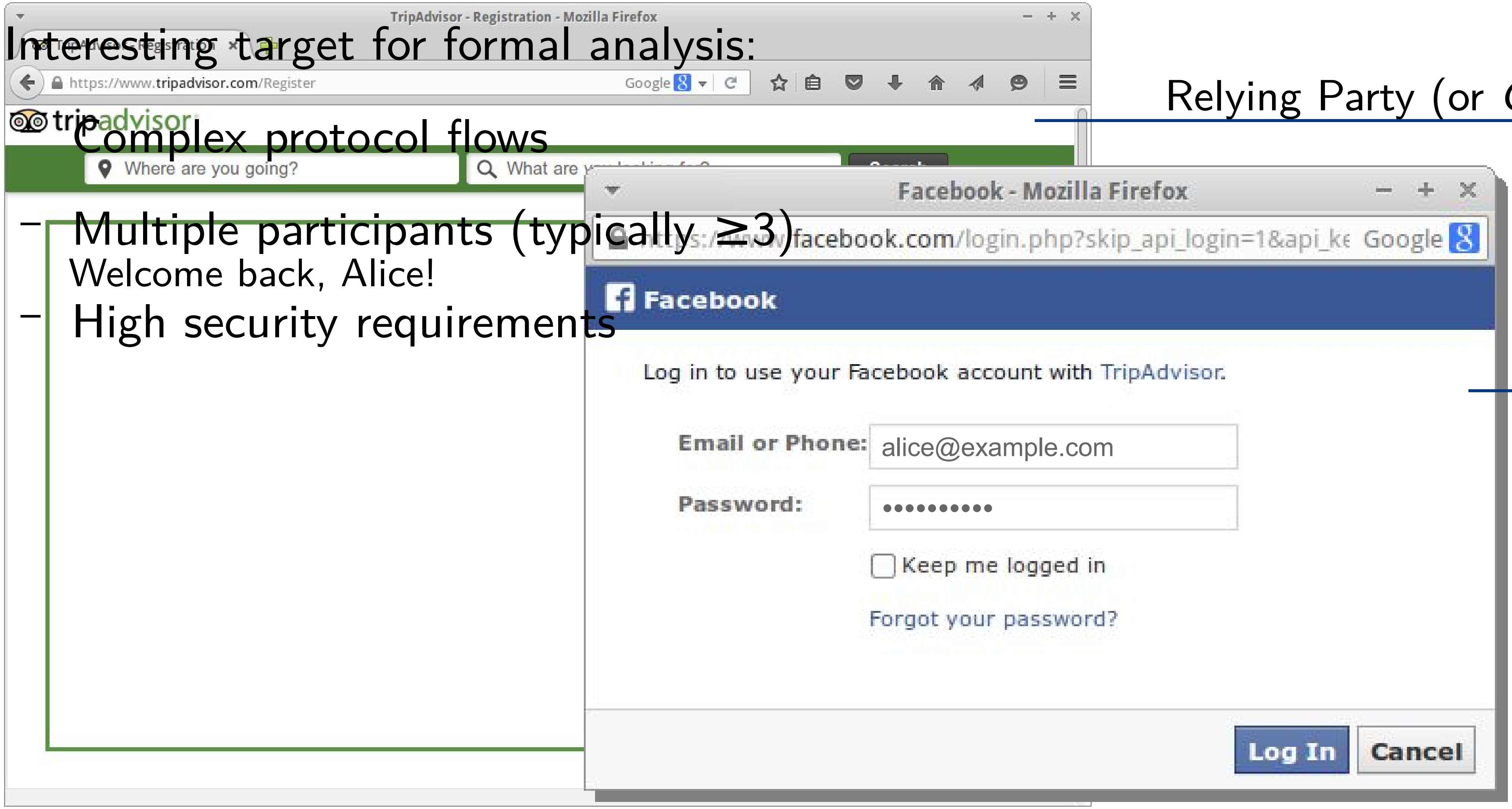


An Expressive Formal Model of the Web Infrastructure



WIM Case Studies

- Web **single sign-on (SSO)** systems
- Interesting target for formal analysis:
  - Complex protocol flows
  - Multiple participants (typically ≥3)
  - High security requirements



Relying Party (or *Client*)

Identity Provider

WIM Case Studies

SPRESSO
https://spresso.me

Mozilla BrowserID

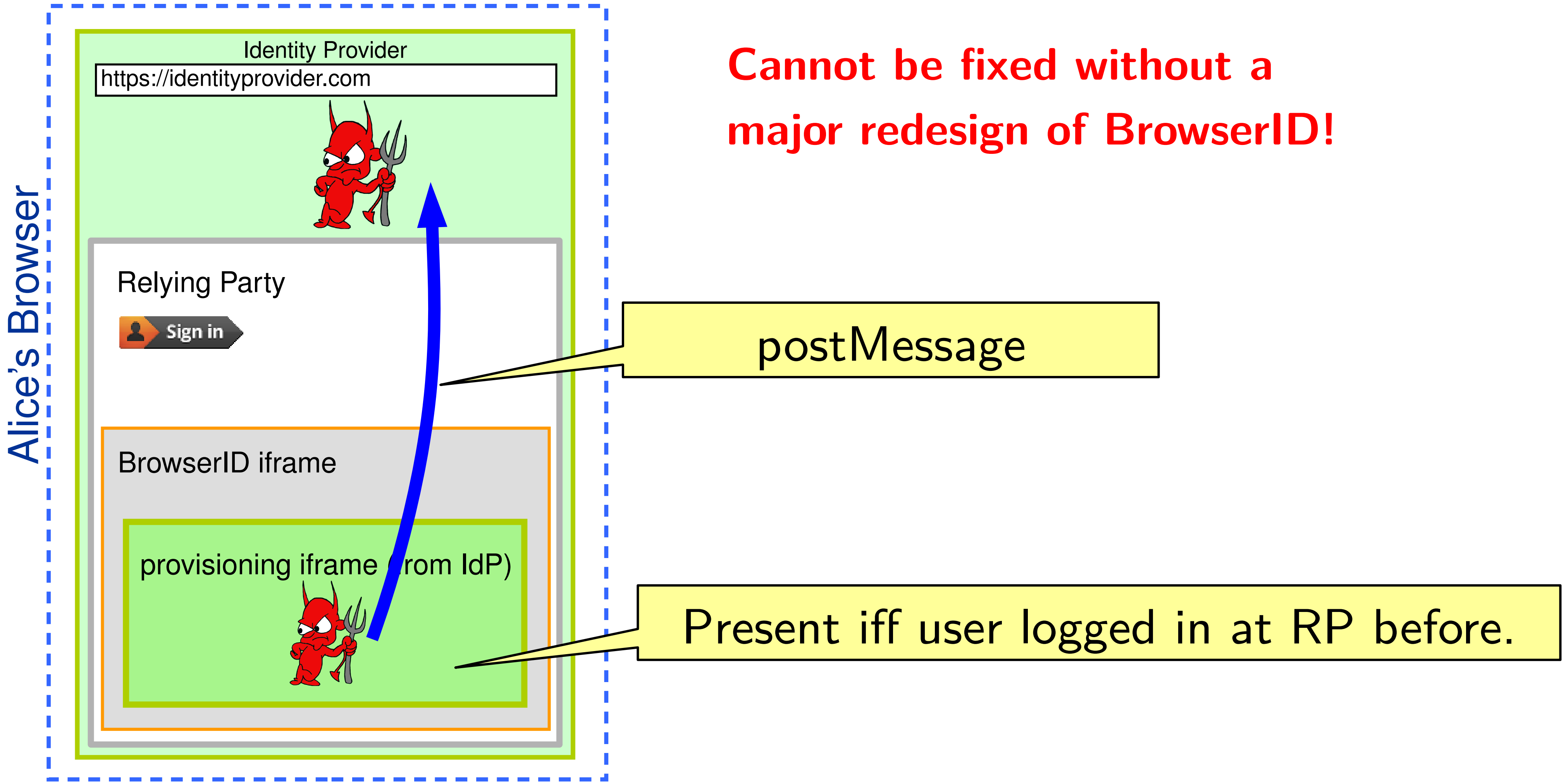
OAuth 2.0

OpenID Connect

- Discovered severe attacks against authentication
- After fixes: Proof of security
- Special feature privacy: broken beyond repair

BrowserID: Privacy Attack

Information is leaked by the **window structure** in the user's browser:



Cannot be fixed without a major redesign of BrowserID!

WIM Case Studies

Mozilla BrowserID

- Discovered severe attacks against authentication
- After fixes: Proof of security
- Special feature privacy: broken beyond repair

SPRESSO

https://spresso.me

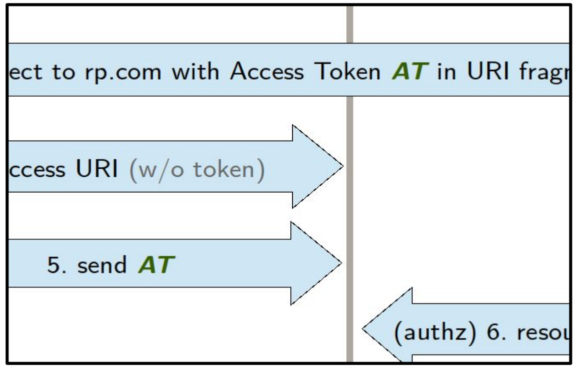
- Designed from scratch
- First formalized in ***WIM***, then implemented
- First SSO with proven privacy and security

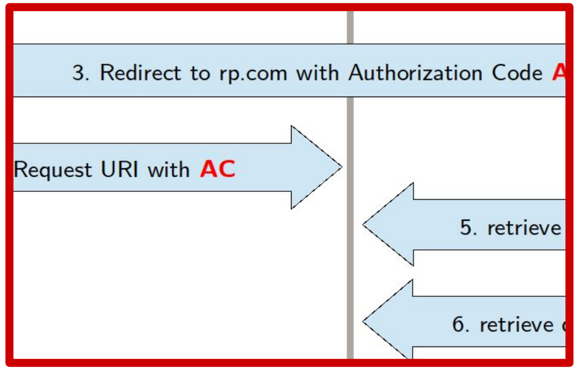
OAuth 2.0

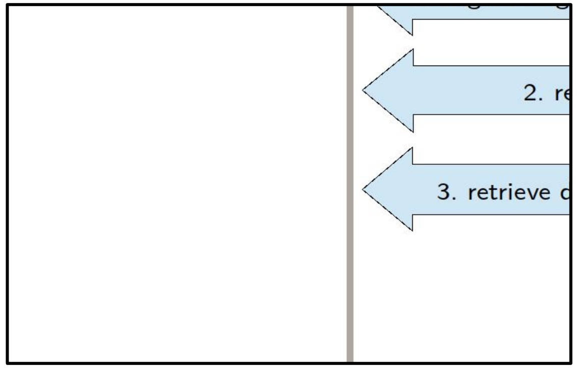
OpenID Connect

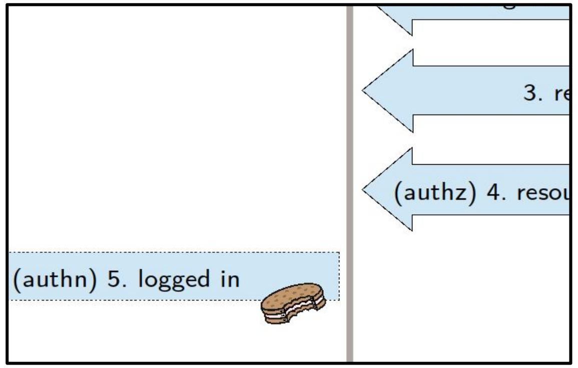
OAuth 2.0

- SSO framework used for authorization/authentication
- Specified by IETF (RFC6749), very widely used (e.g., Log in With Facebook)
- Many "variables":
  optional parameters, *public* and *confidential* clients, etc.
- Four different modes of interaction (*grants*)

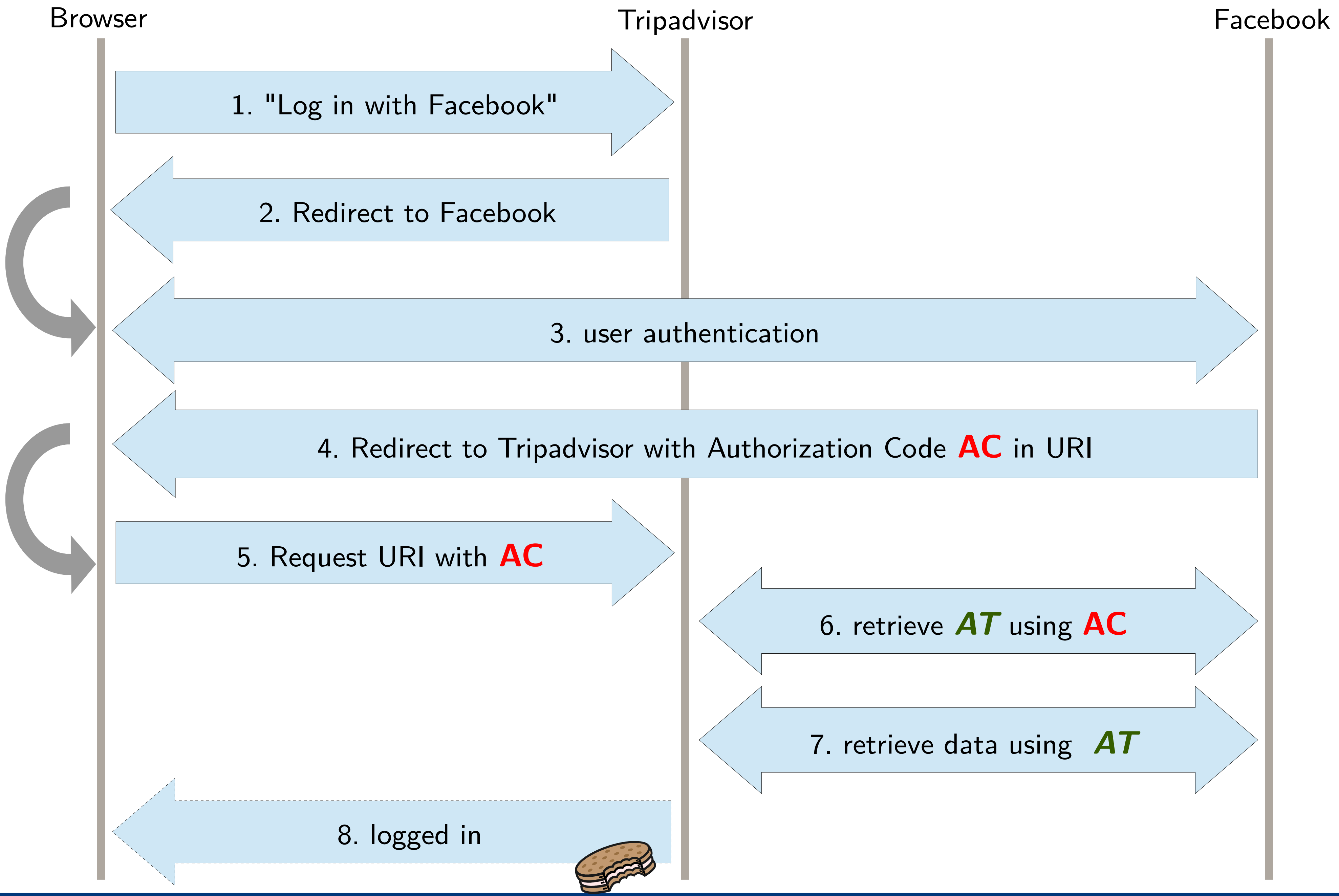




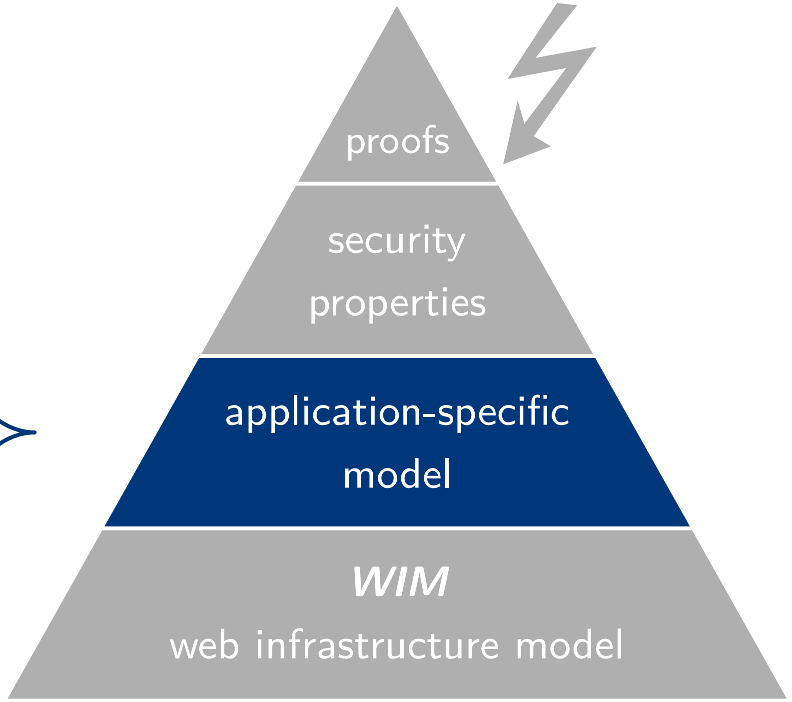




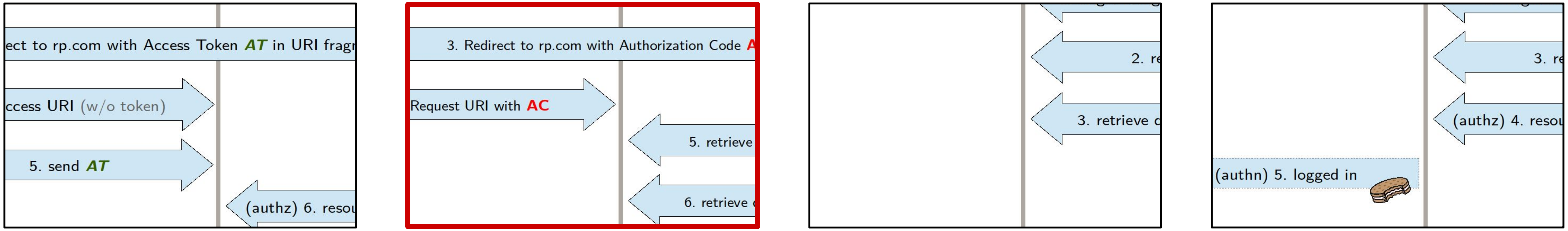
OAuth 2.0


Browser
Tripadvisor
Facebook
1. "Log in with Facebook"
2. Redirect to Facebook
3. user authentication
4. Redirect to Tripadvisor with Authorization Code AC in URI
5. Request URI with AC
6. retrieve AT using AC
7. retrieve data using AT
8. logged in

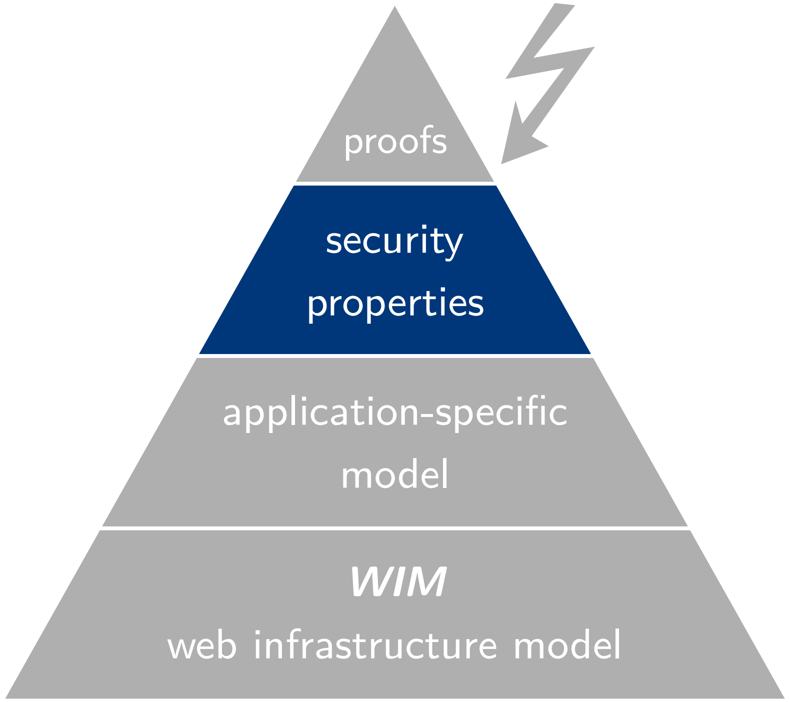
OAuth 2.0



- SSO framework used for authorization/authentication
- Specified by IETF (RFC6749), very widely used (e.g., Log in With Facebook)
- Many "variables": optional parameters, *public* and *confidential* clients, etc.
- Four different modes of interaction (*grants*)



OAuth 2.0: Security Properties



- Authentication

  Definition 56 (Authentication Property). Let $\mathcal{OAuthWS}^n$ be an OAuth web system with a network attacker. We say that $\mathcal{OAuthWS}^n$ *is secure w.r.t. authentication* iff for every run ρ of $\mathcal{OAuthWS}^n$, every state (S^j, E^j, N^j) in ρ, every $r \in \mathsf{Clients}$ that is honest in S^j, every $i \in \mathsf{OAP}$, every $g \in \mathsf{dom}(i)$, every $u \in \mathbb{S}$, every client service token of the form $\langle n, \langle u, g \rangle \rangle$ recorded in $S^j(r).\mathtt{serviceTokens}$, and n being derivable from the attackers knowledge in S^j (i.e., $n \in d_\emptyset(S^j(\mathsf{attacker}))$), then the browser b owning u is fully corrupted in S^j (i.e., the value of $isCorrupted$ is $\mathtt{FULLCORRUPT}$), some $r' \in \mathsf{trustedClients}(\mathsf{secretOfID}(\langle u, g \rangle))$ is corrupted in S^j, or i is corrupted in S^j.

- Authorization

  Definition 55 (Authorization Property). Let $\mathcal{OAuthWS}^n$ be an OAuth web system with a network attacker. We say that $\mathcal{OAuthWS}^n$ *is secure w.r.t. authorization* iff for every run ρ of $\mathcal{OAuthWS}^n$, every state (S^j, E^j, N^j) in ρ, every OAP $i \in \mathsf{OAP}$, every $r \in \mathsf{Clients} \cup \{\bot\}$ with r being honest in S^j unless $r = \bot$, every $u \in \mathsf{ID} \cup \{\bot\}$, for $n = \mathsf{resourceOf}(i, r, u)$, n is derivable from the attackers knowledge in S^j (i.e., $n \in d_\emptyset(S^j(\mathsf{attacker}))$), it follows that

  1. i is corrupted in S^j, or
  2. $u \neq \bot$ and (i) the browser b owning u is fully corrupted in S^j or (ii) some $r' \in \mathsf{trustedClients}(\mathsf{secretOfID}(u))$ is corrupted in S^j.

OAuth 2.0: Security Properties

- Session Integrity for authentication

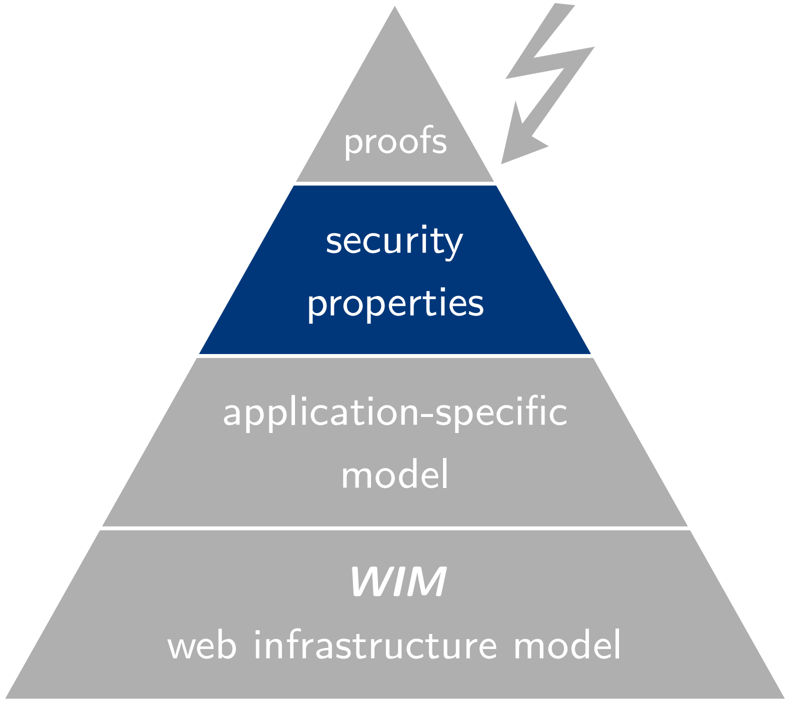
Definition 64 (Session Integrity for Authentication). Let $\mathcal{OA}uth\mathcal{WS}^w$ be an OAuth web system with web attackers. We say that $\mathcal{OA}uth\mathcal{WS}^w$ *is secure w.r.t. session integrity for authentication* iff for every run ρ of $\mathcal{OA}uth\mathcal{WS}^w$, every processing step Q_{login} in ρ, every browser b that is honest in Q_{login}, every $r \in \mathsf{Clients}$ that is honest in Q_{login}, every $i \in \mathsf{OAP}$, every identity $\langle u, g \rangle$, the following holds true: If in Q_{login} a service token of the form $\langle n, \langle \langle u', g' \rangle, m \rangle \rangle$ for a domain $m \in \mathsf{dom}(i)$ and some n, u', g' is created in r (in Line 38 of Algorithm B.4) and n is sent to the browser b, then

(a) there is an OAuth Session $o \in \mathsf{OASessions}(\rho, b, r, i)$, and

(b) if i is honest in Q_{login} then Q_{login} is in o and we have that

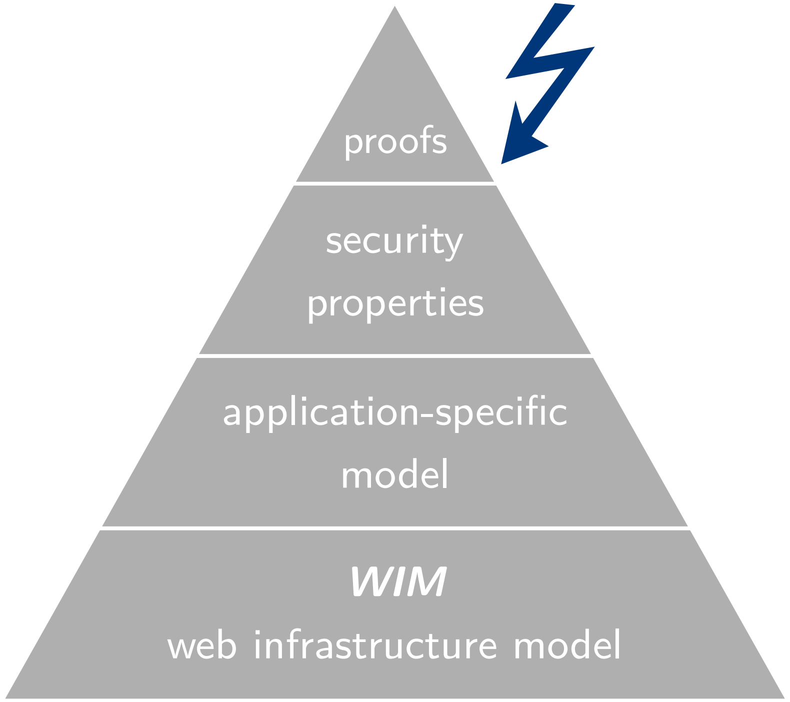
$$\big(\mathsf{selected}_{\text{ia}}(o, b, r, \langle u, g \rangle) \vee \mathsf{selected}_{\text{nia}}(o, b, r, \langle u, g \rangle)\big) \iff \big(\langle u, g \rangle \equiv \langle u', g' \rangle\big)\ .$$

- Session Integrity for authorization (similar to above)



OAuth 2.0: New Attacks

OAuth 2.0 had been analyzed many times before, but not in a comprehensive formal model.



Further Related Work (OAuth 2.0)

- [Bansal et al., 2012-2014]
- [Wang et al., 2013]
  - "Explicating SDKs"
  - Boogie/Corral
  - Extraction of SDK logic, definition of security properties, addition of assume statements, code verification.
- [Chari, Jutla, Roy, 2011]
  - UC model analysis of OAuth Authorization Code Grant
  - No web features
- Several empirical studies, focussed on typical implementation errors

Further Related Work (OpenID Connect)

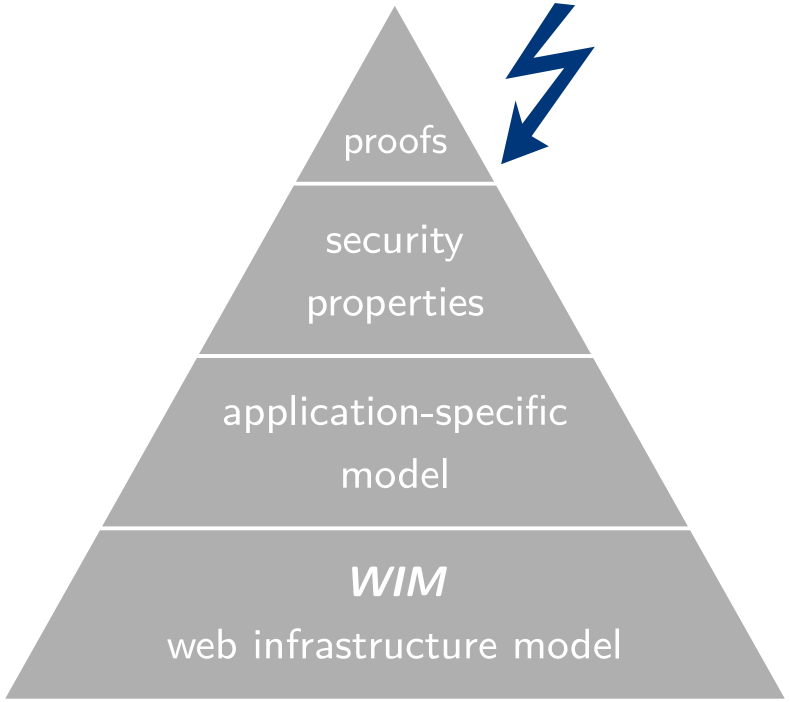
- [Mladenov et al., 2016]
  - Specific variant of the IDP Mix-Up attack
  - No formal model
- [Li, Mitchell, 2016]
  - Implementation errors in deployments of Google Sign-In

OAuth 2.0: New Attacks

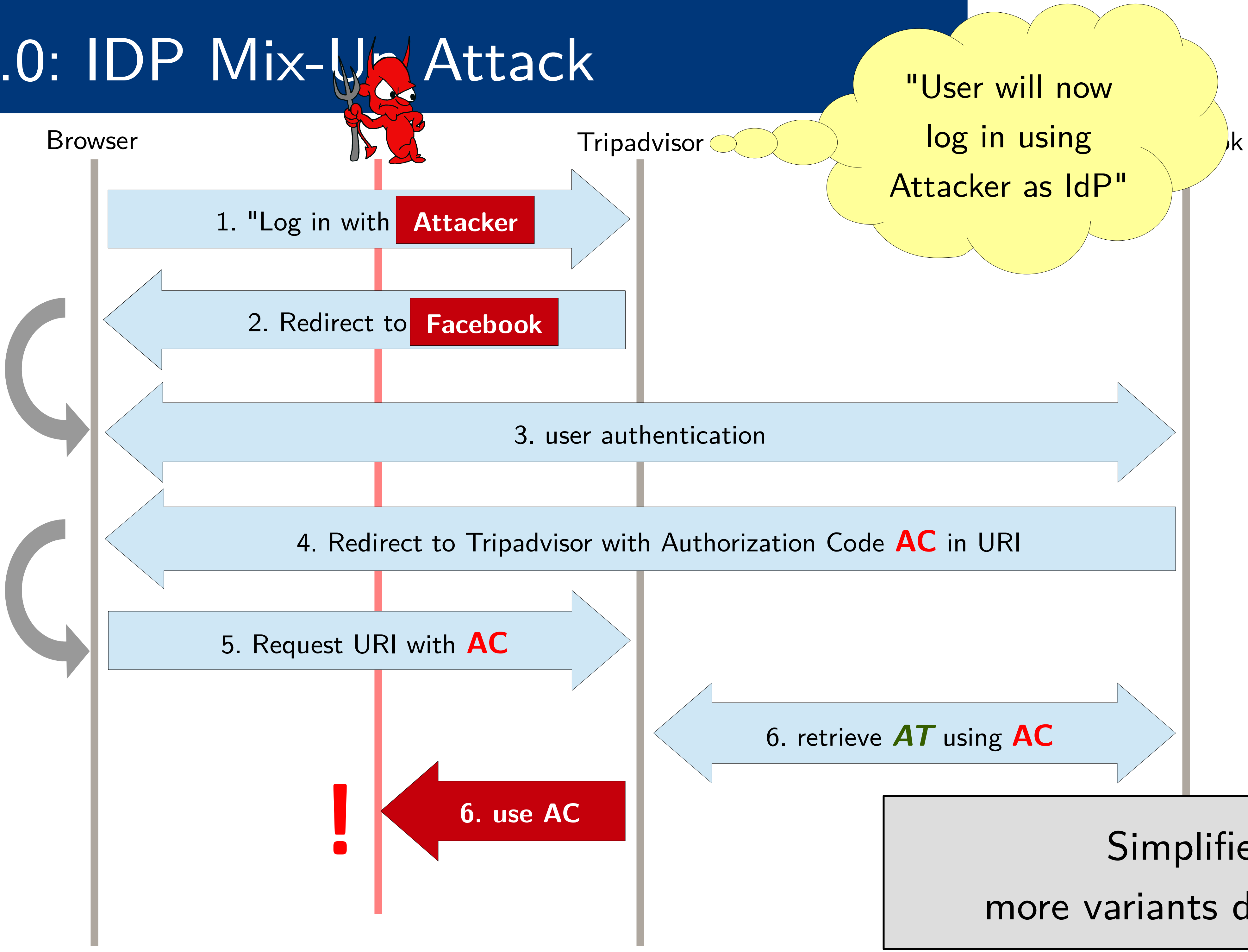
OAuth 2.0 had been analyzed many times before, but not in a comprehensive formal model.

New attacks:

- 307 Redirect Attack
- Identity Provider Mix-Up Attack (new class of attacks)
- State Leak Attack
- Naïve Client Session Integrity Attack
- Across Identity Provider State Reuse Attack



OAuth 2.0: IDP Mix-Up Attack



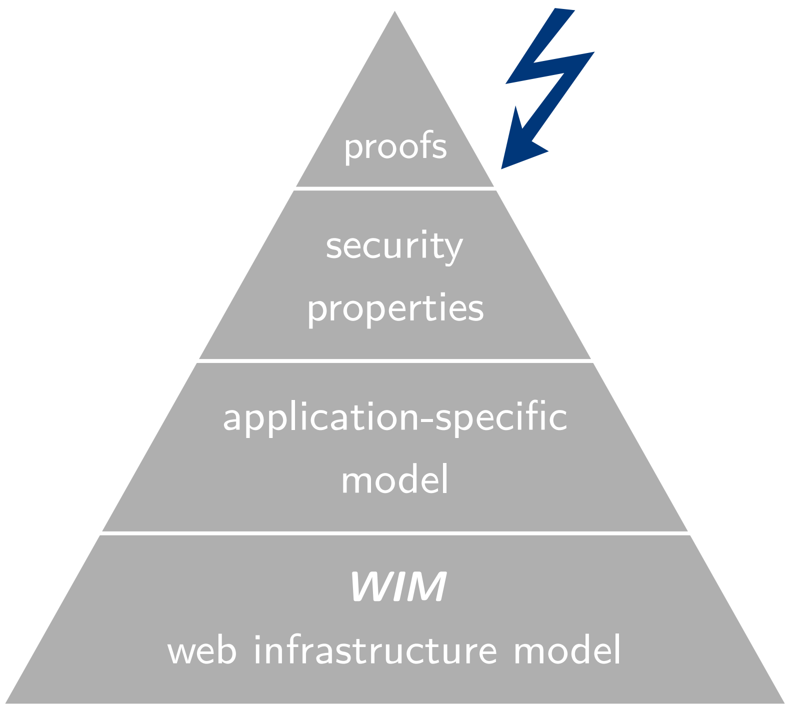
Simplified,
more variants discovered

OAuth 2.0: New Attacks

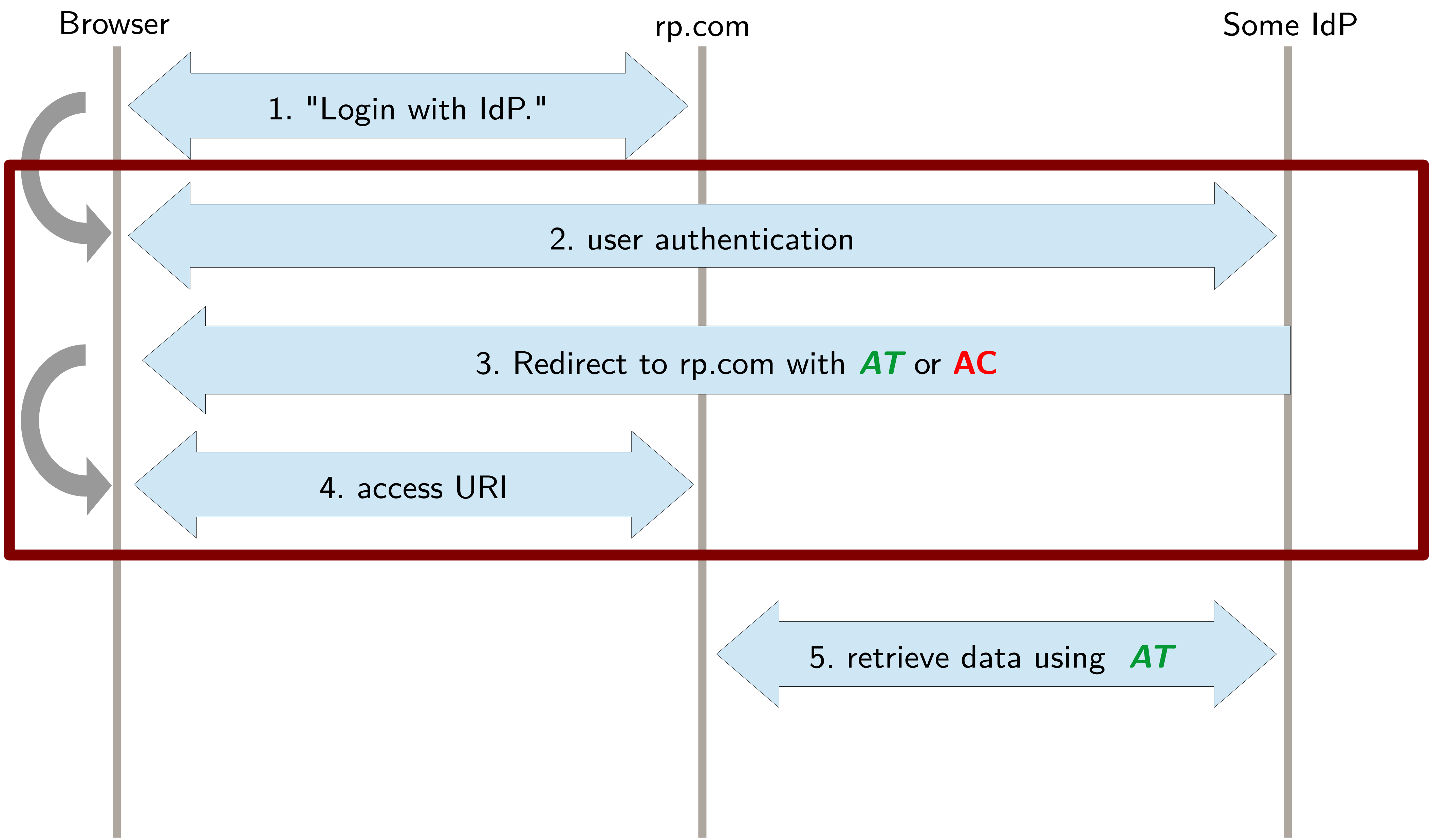
OAuth 2.0 had been analyzed many times before, but not in a comprehensive formal model.

New attacks:

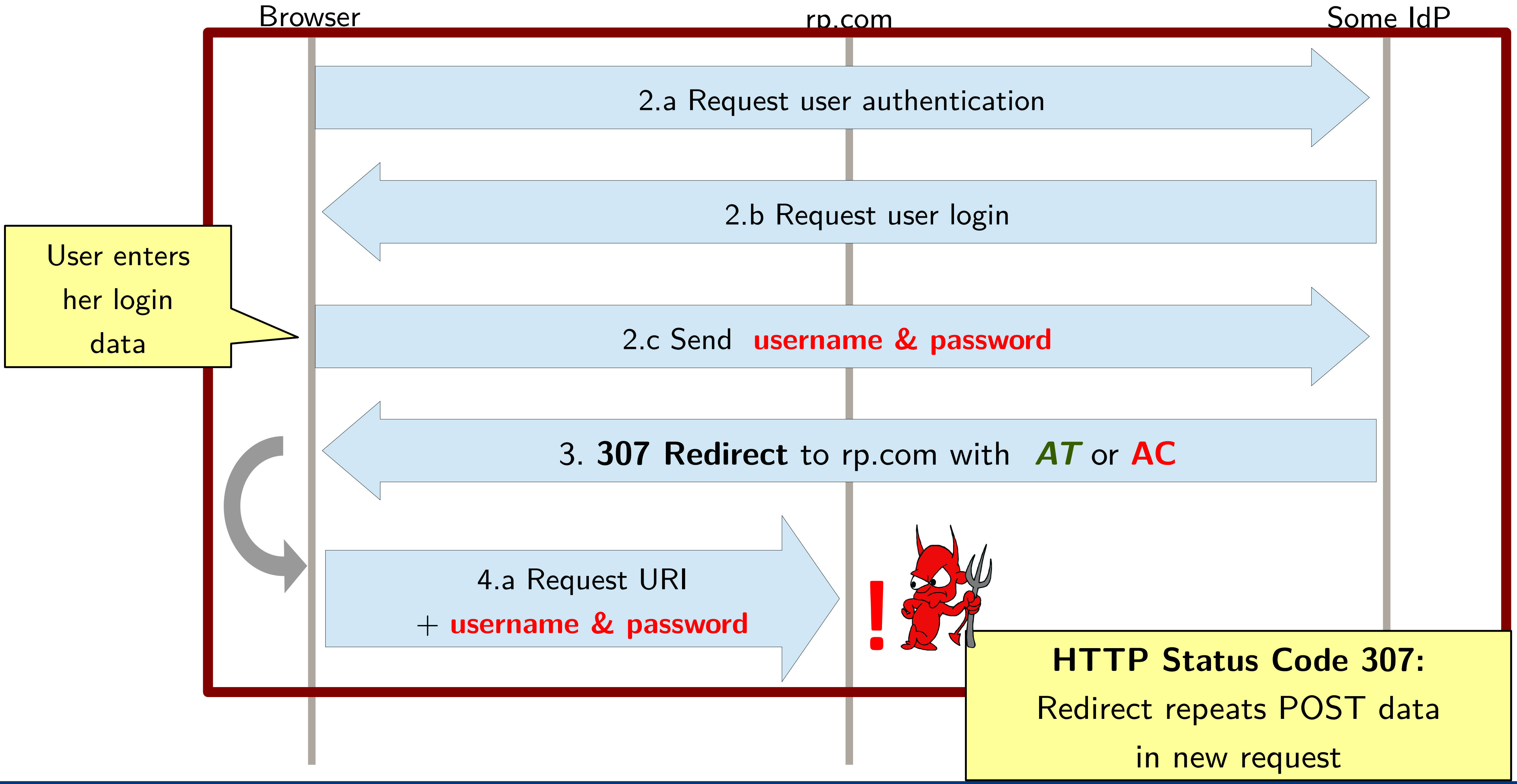
- 307 Redirect Attack
- Identity Provider Mix-Up Attack (new class of attacks)
- State Leak Attack
- Naïve Client Session Integrity Attack
- Across Identity Provider State Reuse Attack



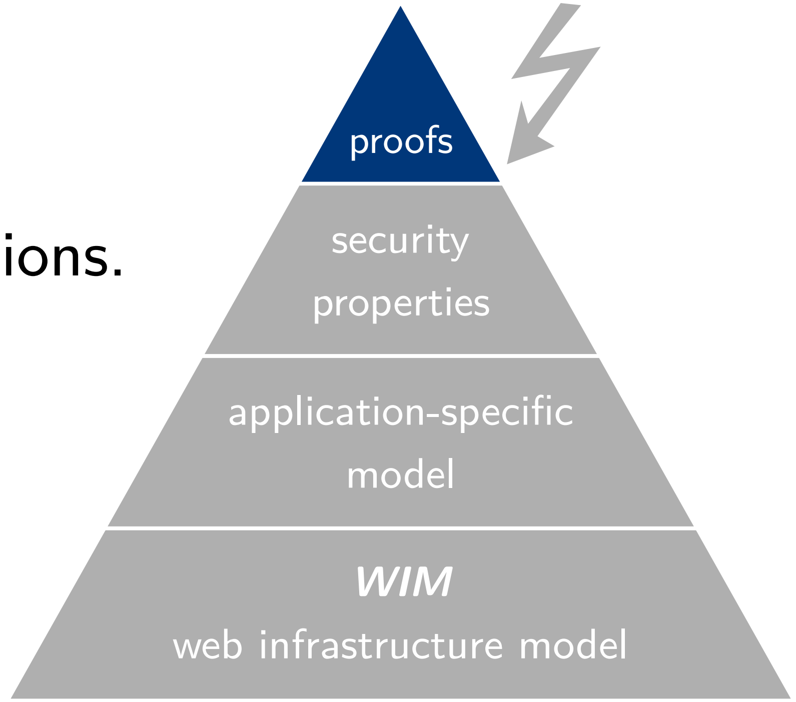
OAuth: 307 Redirect Attack (I)



OAuth: 307 Redirect Attack (II)



OAuth 2.0: Proof of Security



Proof based on our model of OAuth 2.0 with all grant types and options.

Assumptions:

- Adherence to web best practices (e.g., regarding session handling)
- Adoption of our implementation guidelines (e.g., no 3rd party scripts on certain web pages)
- Fixes against previously known and new attacks

Theorem 1. Let $\mathcal{OAuthWS}^n$ be an OAuth web system with a network attacker, then $\mathcal{OAuthWS}^n$ is secure w.r.t. authorization and secure w.r.t. authentication. Let $\mathcal{OAuthWS}^w$ be an OAuth web system with web attackers, then $\mathcal{OAuthWS}^w$ is secure w.r.t. session integrity for authorization and authentication.

OAuth 2.0: Impact

- Disclosed OAuth attacks to the IETF Web Authorization Working Group in late 2015
- Emergency meeting with the working group four weeks later
- Initiated the OAuth Security Workshop (OSW) to foster the exchange between researchers, standardization groups, and industry
- Joined the working group to codify the fixes into a new RFC:
  OAuth 2.0 Security Best Current Practice
  [draft-ietf-oauth-security-topics]

WIM Case Studies

Mozilla BrowserID

- Discovered severe attacks against authentication
- After fixes: Proof of authentication
- Special feature privacy: broken beyond repair

SPRESSO

https://spresso.me

- Designed from scratch
- First formalized in ***WIM***, then implemented
- First SSO with proven privacy and security

OAuth 2.0

OpenID Connect

- Found several new attacks
- Developed fixes and implementation guidelines
- Proof of security

OpenID Connect

- OAuth 2.0 was built for authorization, not authentication
- OpenID Connect: "Identity Layer" for OAuth 2.0 to solve this
- Includes new extensions:
  - Automatic discovery of identity providers
  - Dynamic registration of clients at identity providers

  } Out of scope of plain OAuth 2.0
- New token type ("id token")
- Cryptographic mechanisms, e.g., signed id token

OpenID Connect

Results:

- All newly discovered OAuth attacks apply to OpenID Connect as well
- Implementation guidelines to avoid known attacks
- Proof of security (authentication, authorization, session integrity) including discovery and dynamic registration extensions

Theorem 2 (Security of OpenID Connect). Let $\mathcal{OIDCWS}^n$ be an OIDC web system with a network attacker. Then, $\mathcal{OIDCWS}^n$ is secure w.r.t. authentication and authorization. Let $\mathcal{OIDCWS}^w$ be an OIDC web system with web attackers. Then, $\mathcal{OIDCWS}^w$ is secure w.r.t. session integrity for authentication and authorization.

WIM Case Studies

Mozilla BrowserID

- Discovered severe attacks against authentication
- After fixes: Proof of authentication
- Special feature privacy: broken beyond repair

SPRESSO
https://spresso.me

- Designed from scratch
- First formalized in ***WIM***, then implemented
- First SSO with proven privacy and security

OAuth 2.0

- Found several new attacks
- Developed fixes and implementation guidelines
- Proof of security

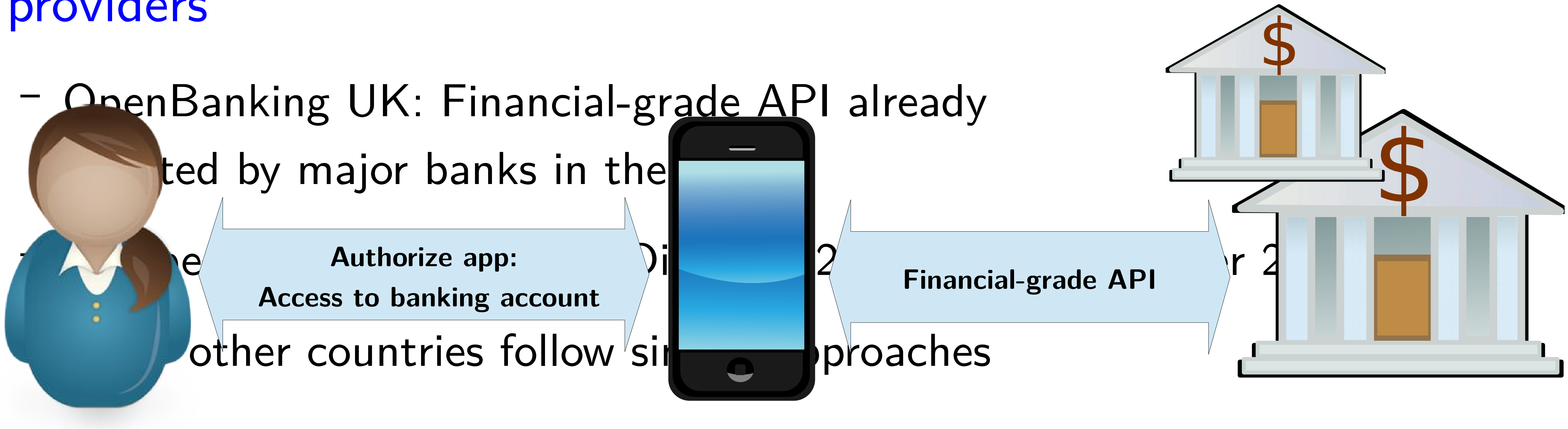
OpenID Connect

- Including extensions
- Developed best practices against known attacks
- Proof of security

Most recent case study: **Financial-grade API (FAPI)**

Motivation FAPI

- Authorization and authentication in high-risk scenarios
- Laws and activities for opening financial services to third-party providers
  - OpenBanking UK: Financial-grade API already … ted by major banks in the
  - … e … Di … 2 … r 2
  - … other countries follow si … proaches

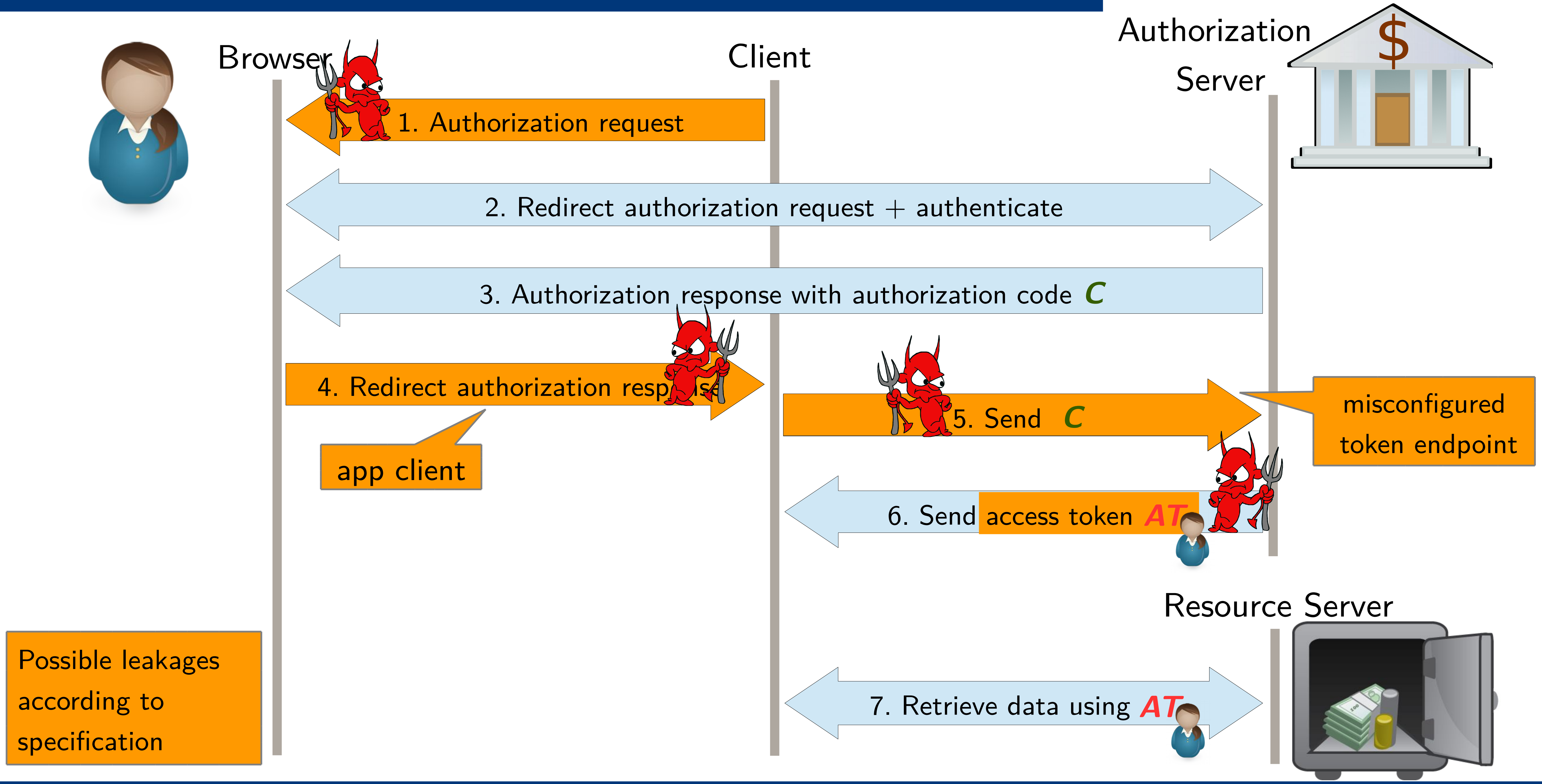


Overview FAPI

OpenID Financial-grade API:

- Hardened version of **OAuth 2.0** for high-risk use-cases
- **New mechanisms:** OAuth 2.0 Token Binding, OAuth 2.0 Mutual TLS, Proof Key for Code Exchange, JWT Secured Authorization Response Mode

FAPI: Attacker Model



Overview FAPI

- OpenID Financial-grade API:
  - Hardened version of **OAuth 2.0** for high-risk use-cases
  - **New mechanisms:** OAuth 2.0 Token Binding, OAuth 2.0 Mutual TLS, Proof Key for Code Exchange, JWT Secured Authorization Response Mode
- Our Work: formal security analysis of the Financial-grade API
  - Formal model of the Financial-grade API based on the **Web Infrastructure Model**
  - Precise definition of **security properties**
  - During formal analysis: **found several attacks** bypassing the new mechanisms
  - **Proof of security** for the fixed Financial-grade API
- Collaborating with OpenID Foundation to fix the standard

WIM Case Studies

Mozilla BrowserID

- Discovered severe attacks against authentication
- After fixes: Proof of authentication
- Special feature privacy: broken beyond repair

SPRESSO
https://spresso.me

- Designed from scratch
- First formalized in ***WIM***, then implemented
- First SSO with proven privacy and security

OAuth 2.0

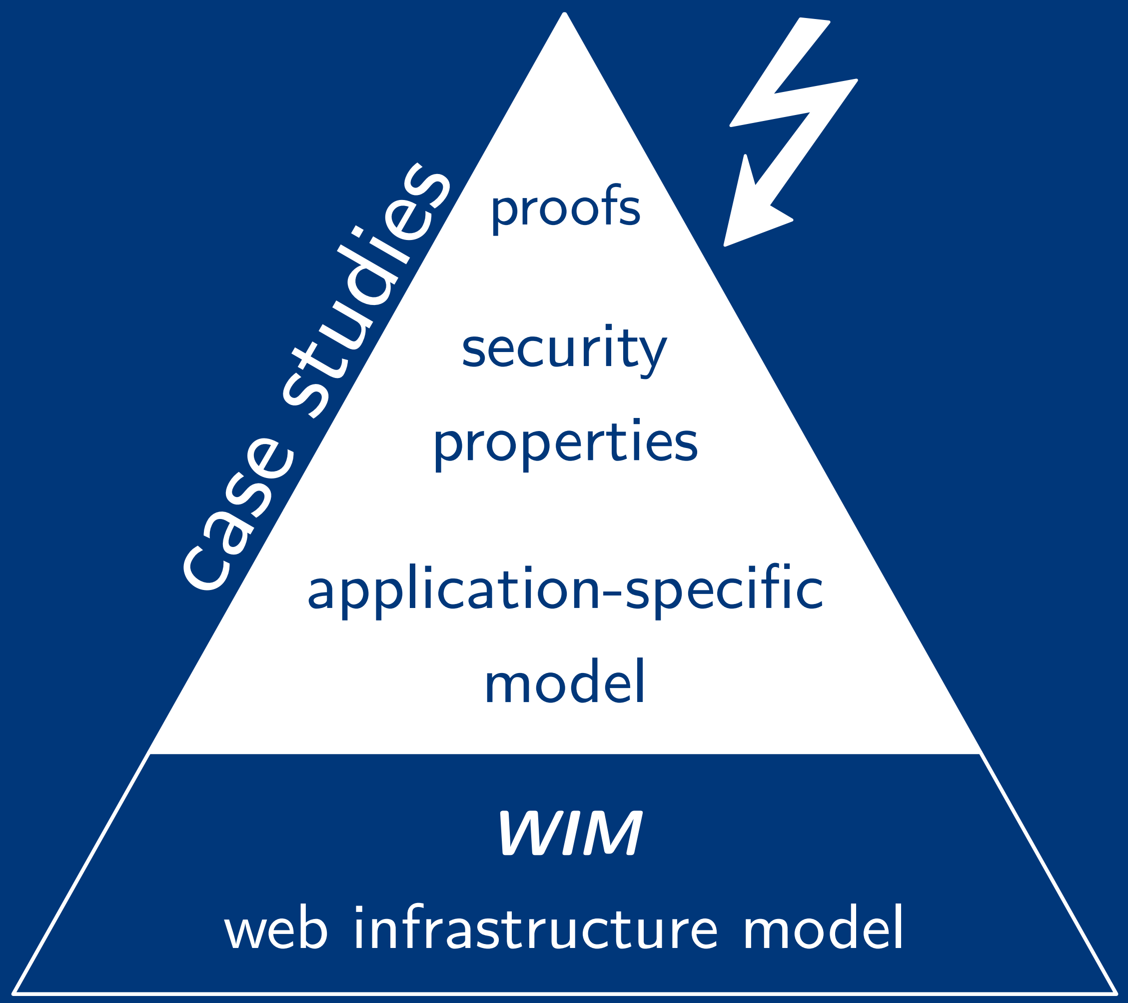
- Found several new attacks
- Developed fixes and implementation guidelines
- Proof of security

OpenID Connect

- Including extensions
- Developed best practices against known attacks
- Proof of security

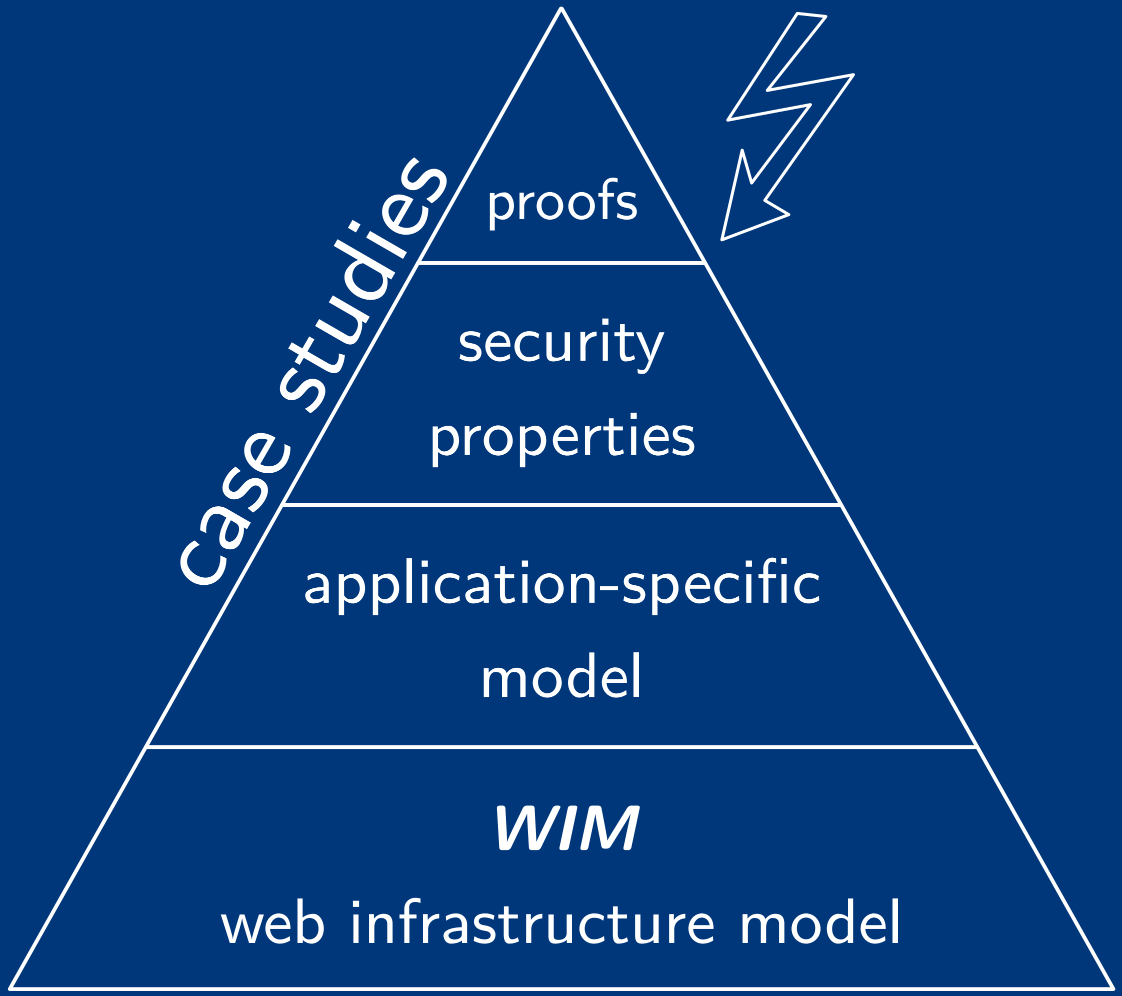
Most recent case study: **Financial-grade API (FAPI)**

An Expressive Formal Model of the Web Infrastructure



WIM: An Expressive Formal Model of the Web Infrastructure

Thank you!



- Most detailed and comprehensive formal model of the web infrastructure so far
- Case studies with real-world impact
- New classes of attacks
- Formal proofs of web security with very high level of detail
- Designed first privacy-preserving SSO system: SPRESSO
- Currently: mechanized model, in collaboration with Bhargavan et al.