Formal Security Analysis of Web Standards and Applications

Prof. Dr. Ralf Küsters

2019/04/03
The Institute of Information Security

Prof. Dr. Ralf Küsters
Head

Dagmar Gahr
Secretary

Pedram Hosseyni, M.Sc.
Ph.D. Student

Julian Liedtke, M.Sc.
Ph.D. Student

Johannes Müller, M.Sc.
Ph.D. Student

Daniel Rausch, M.Sc.
Ph.D. Student

Dipl.-Inf. Guido Schmitz
Ph.D. Student

Dipl.-Math. Mike Simon
Ph.D. Student

Tim Württele, M.Sc.
Ph.D. Student

Dr. Quoc Huy Do
Postdoc

Dipl.-Ing. (FH) Maria
Unger-Zimmermann
System Administrator

Daniel Bernau, M.Sc.
External Ph.D. Student

Benjamin Weggenmann, M.Sc.
External Ph.D. Student

Since 2017
Teaching at SEC

► Lecture on Foundations of Information Security/Cyber Security
  - Intro to Crypto,
  - (In-)Security of Network Protocols and Cryptographic Protocols
  - (Problems of) PKIs
  - Access Control (Linux, SELinux, Android)

► Lecture on System and Web Security
  - Hacking class

► Advanced lectures on
  - Security (e.g., automated analysis)
  - Privacy (e.g., Zero-Knowledge Proofs, Secure Multi-Party Computation)
  - Modern Cryptography (e.g., provable security)

► Student Projects (Hacking Projects) and Seminars

Plus classes on security by colleagues, e.g., Hardware Security.
Research at SEC

► **Applied Cryptography**: security/cryptographic protocols
- Design and analysis
- Authentication, key exchange, privacy-preserving ML/data analysis, etc.
- Provable security, also using and building tools.

► **Web Security**
- Rigorous and systematic formal analysis of web standards and applications (also tool-based)
- Example: Single Sign-On (OAuth 2.0, OpenID Connect, SPRESSO, BrowserID, etc.)

► **Blockchains**

► **E-Voting**

IBM HSM

New attack:

extract master key for deriving PINs.
Many Web Attacks...

Cross-Site Request Forgery
Leaks of Sensitive Data
Clickjacking
Man-in-the-middle Attacks

Attacks on Single Sign-on
DNS Rebinding
Cross-Site Scripting

Finding vulnerabilities is hard!

SQL Injection
Malicious iframes
Missing Checks
Other Injection Attacks
Attacks on Cookies
Attacks on Session Management
Malicious CDNs

Finding vulnerabilities is hard!
Current Methods

Expert review
of standards and implementations

CHECKLIST

- CSRF
- Session Swapping
- Missing Checks
- Cross-Origin Attacks
- Insecure Connection
- Man-in-the-middle

Penetration testing
using tools or manual analysis

We have had several Bachelor and Master theses on this with local companies.
(REST APIs, ECUs, ...)
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 attacks?
(And even have security guarantees?)
Our Model-Based Approach

Foundation:
Formal description of the web

Precise Formal Security Properties

Application-specific model

Generic web infrastructure model (WIM)

Formal Proofs of Properties

Attacks

 Fixes

© Rinse and repeat until proof goes through.

Application model built from source code or specification
Advantages

This approach can yield...

- **new (classes of) attacks** and respective fixes
- strong **security guarantees** excluding even unknown types of attacks
An Expressive Formal Model of the Web Infrastructure

WIM

web infrastructure model

application-specific model

security properties

proofs

[S&P2014]
[ESORICS2015]
[CCS2015]
[CCS2016]
[CSF2017]
[SP2019]
An Expressive Formal Model of the Web Infrastructure

proofs
security properties
application-specific model

WIM
web infrastructure model

[S&P2014]
[ESORICS2015]
[CCS2015]
[CCS2016]
[CSF2017]
[SP2019]
WIM Network Model and Attackers
Including ...

- DNS, HTTP, HTTPS
- window & document structure
- scripts
- attacker scripts
- web storage & cookies
- web messaging & XHR
- message headers
- redirections
- security policies
- dynamic corruption
- WebRTC
- ...

Origin: https://example.com
WIM Web Browser Model - Example

Algorithm 8 Web Browser Model: Process an HTTP response.

1: function PROCESSRESPONSE(response, reference, request, requestUrl, key, f, f')
2:    if Set-Cookie ∈ response.headers then
3:        for each c ∈ {0} response.headers[Set-Cookie], c ∈ Cookies do
4:            let f'.cookies[request.host] ← := AddCookie(f'.cookies[request.host], c)
5:    if Strict-Transport-Security ∈ response.headers ∧ requestUrl.protocol ≡ S then
6:        let f'.sts := f'.sts + {0} request.host
7:    if Referer ∈ request.headers then
8:        let referrer := request.headers[Referer]
9:    else
10:       let referrer := ⊥
11:    if Location ∈ response.headers ∧ response.status ∈ {303, 307} then
12:       let url := response.headers[Location]
13:    if url.fragment ≡ ⊥ then
14:       let url.fragment := requestUrl.fragment
15:    let method' := request.method
16:    let body' := request.body
17:    if Origin ∈ request.headers then
18:       let origin := ⟨request.headers[Origin], ⟨request.host, url.protocol⟩⟩
19:    else
20:       let origin := ⊥
21:    if response.status ≡ 303 ∧ request.method ∉ {GET, HEAD} then
22:       let method' := GET
23:       let body' := {}
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

- Proofs
- Security properties
- Application-specific model

WIM (Web Infrastructure Model)

References:
- [S&P2014]
- [ESORICS2015]
- [CCS2015]
- [CCS2016]
- [CSF2017]
- [SP2019]
An Expressive Formal Model of the Web Infrastructure

proofs
security properties
application-specific model

WIM
web infrastructure model

[S&P2014]
[ESORICS2015]
[CCS2015]
[CCS2016]
[CSF2017]
[SP2019]
Single Sign-On (SSO)

Welcome back, Alice!

OAuth 2.0
OpenID Connect
Always discovered new and severe attacks
(even when standards have been intensively looked at and analyzed before)

Proposed fixes

Proved security in WIM: this provably excludes a large class of attacks.

Designed own provably secure and privacy-preserving SSO system:

https://spresso.me
OAuth 2.0: Security Definitions

- **Authentication**
  
  Attacker cannot log in at relying party with honest identity

- **Authorization**
  
  Attacker cannot access resources of honest identity

- **Session Integrity**
  
  Honest user is logged in under her own account and using her own resources
Definition 46 (Authentication Property). Let $OWS^n$ be an OAuth web system with a network attacker. We say that $OWS^n$ is secure w.r.t. authentication iff for every run $\rho$ of $OWS^n$, every state $(S^j, E^j, N^j)$ in $\rho$, every $r \in$ RP that is honest in $S^j$, every $i \in$ IDP, every $g \in$ dom$(i)$, every $u \in S$, every RP service token of the form $\langle n, \langle u, g \rangle \rangle$ recorded in $S^j(r)\.serviceTokens$, and $n$ being derivable from the attackers knowledge in $S^j$ (i.e., $n \in d_\emptyset(S^j($attacker$)))$, then the browser $b$ owning $u$ is fully corrupted in $S^j$ (i.e., the value of isCorrupted is FULLCORRUPT), some $r' \in$ trustedRPs(secretOfID$(\langle u, g \rangle))$ is corrupted in $S^j$, or $i$ is corrupted in $S^j$. 
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

*We also proposed fixes.*
Impact

- Disclosed OAuth attacks to the IETF Web Authorization Working Group in late 2015
- Since then: In close contact with the IETF and OpenID Foundation to improve standards
- Initiated the OAuth Security Workshop (OSW) to foster the exchange between researchers, standardization groups, and industry
- Working together with the IETF for creating new RFC: OAuth 2.0 Security Best Current Practice [draft-ietf-oauth-security-topics]
- We improved standard(s) for millions of users.
WIM Case Studies: SSO

- Always discovered new and severe attacks
  (even when standards have been intensively looked at and analyzed before)
- Proposed Fixes
- Proved security in WIM: this provably excludes a large class of attacks.

Designed own provably secure and privacy-preserving SSO system:

SPRESSO
https://spresso.me
Research at SEC

► **Cryptography**: security/cryptographic protocols
  - Design and analysis
  - Authentication, key exchange, privacy-preserving ML/data analysis, etc.
  - Provable security, also tool-based.

► **Web Security**
  - Rigorous and systematic formal analysis of web standards and applications
    (also tool-based)
  - Example: Single Sign-On (OAuth 2.0, OpenID Connect, SPRESSO, BrowserID, etc.)

► **Blockchains**

► **E-Voting**
My Vision

► Do something like what we did in the web for automotive security
  - Security model of (parts of) modern vehicles
  - Precise attacker model
  - Precise security and privacy properties
  - Systematic security and privacy design and analysis

► Automotive Security Research Center at University of Stuttgart
  - Automotive security research and expertise concentrated in one place
  - Make it easy to collaborate
  - Do the above and other things

Interested? Talk to me.