

AuthScan: Automatic Extraction of Web Authentication Protocols from Implementations

Guangdong Bai¹

Jike Lei¹

Guozhu Meng¹

Sai Sathyanarayan Venkatraman¹

Prateek Saxena¹

Jun Sun²

Yang Liu³

Jin Song Dong¹

¹National University of Singapore

²Singapore University of Technology and Design

³Nanyang Technological University

Web Authentication Schemes & Single Sign-On

- Web Authentication

Sign In Create an Account

Email or Nickname

Password

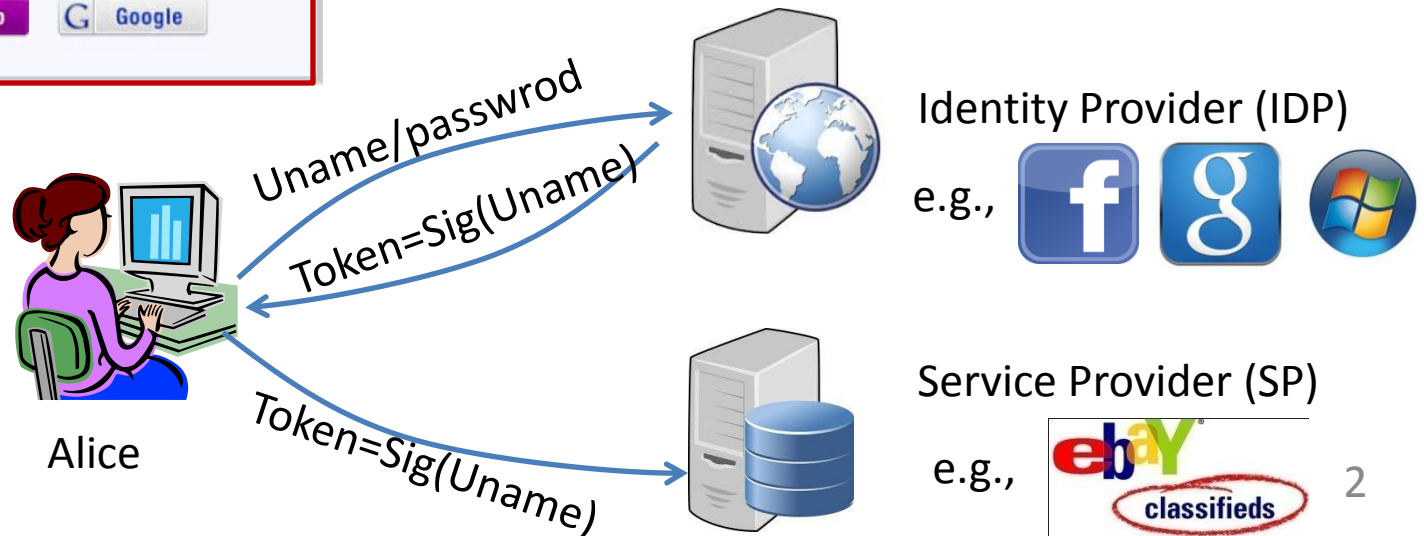
[Forgot your password?](#)

OR

Login with another account:

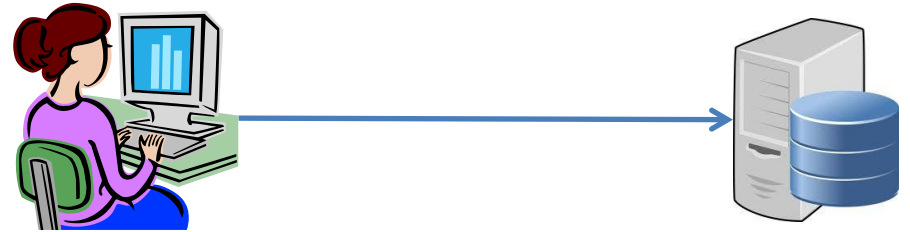
- Single Sign-On (SSO)

- BrowserID (Mozilla)
- Facebook Connect
 - 250+ Million users, 2,000,000 websites
- OpenID
 - one billion users, 50,000 websites
- ...

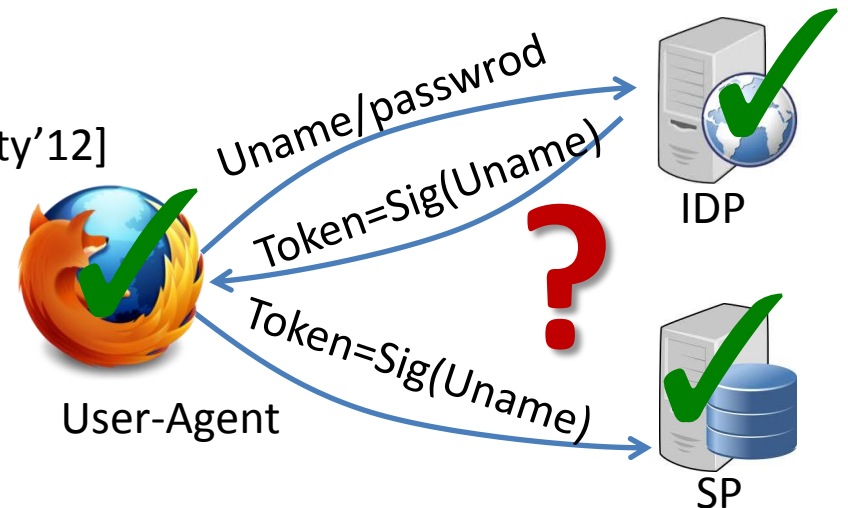


Implementations Can have Bugs!!

- Web Authentication
 - Password Guessing
 - Session/Cookie Stealing
 - ...

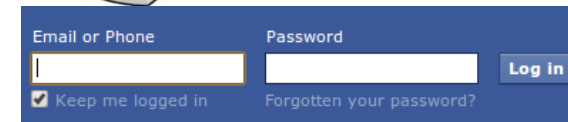


- Harder in SSO implementations
 - Vulnerabilities [BlackHat'07, Oakland'12, CCS'12, USENIX Security'12]



Is Manual Analysis Possible?

- Manual analysis is impractical
 - Closed source
 - Numerous implementations
 - OAuth 1.0 & 2.0: **47** implementations



OAuth (RFC 5849 & RFC 6749)

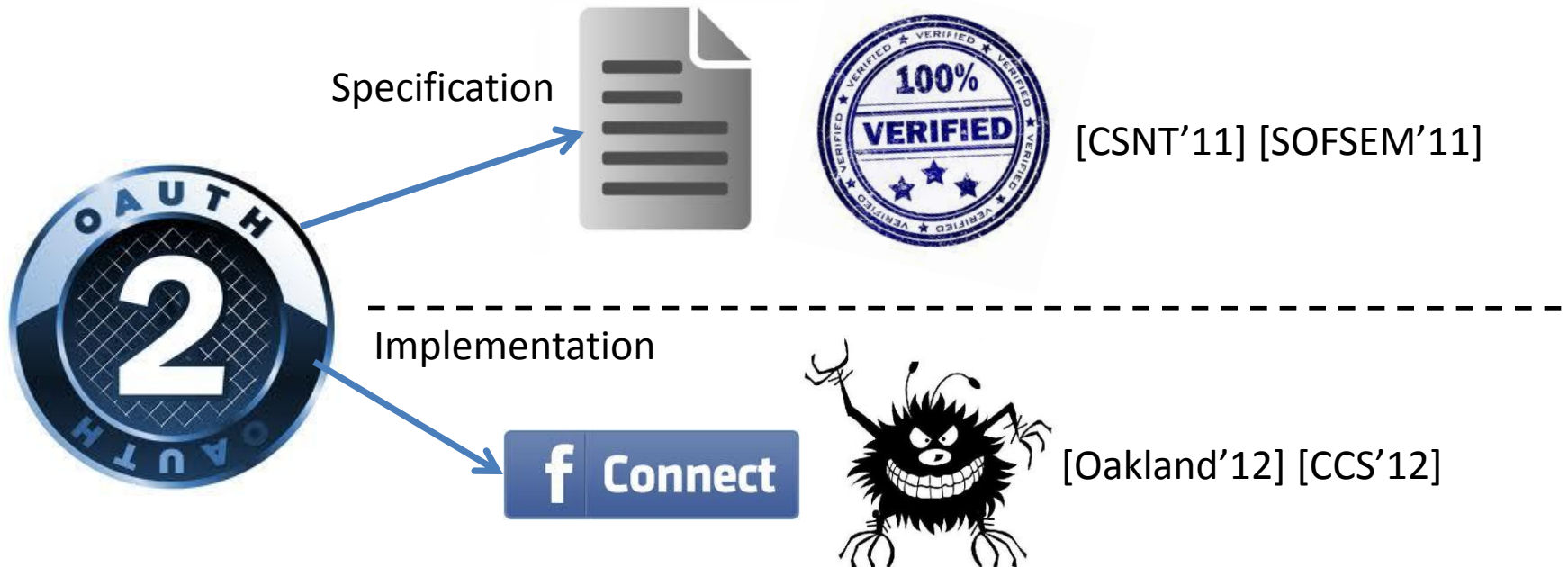


Can't We Verify the Web Authentication?

- Previous protocol verification: **design-level** protocol **specifications**

It is the **IMPLEMENTATION** that security relies on!!

Implementation == Specification?



Our Solution & Contributions

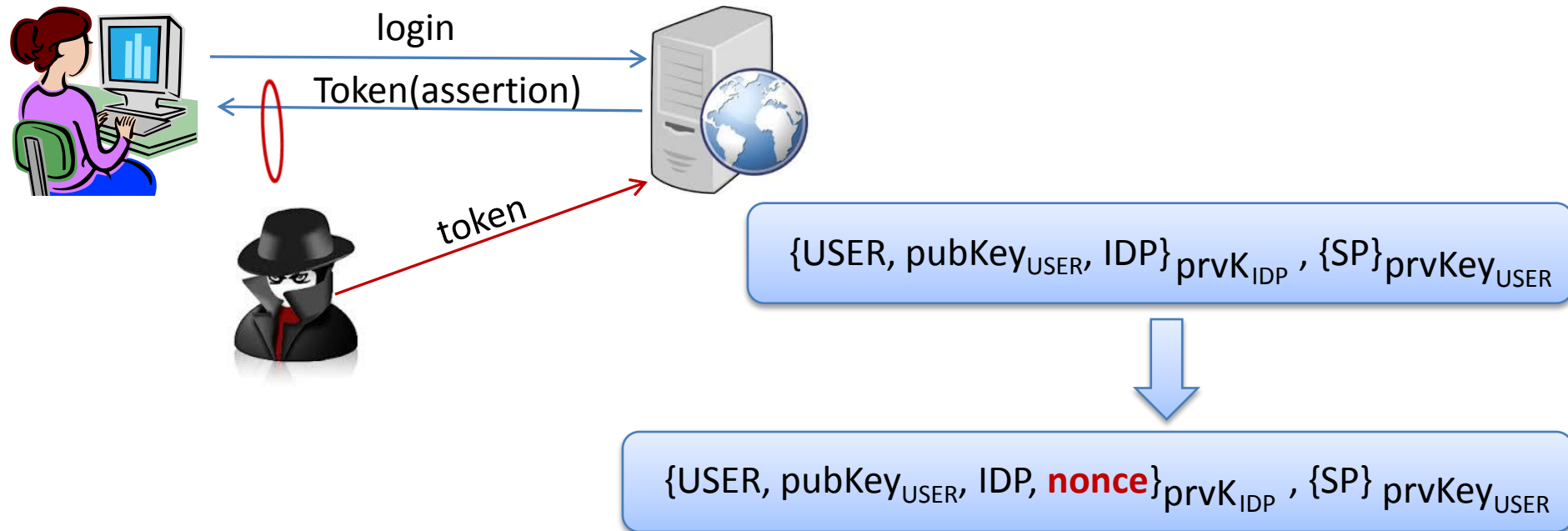
#1 Automatically extract protocols **from implementations**

#2 Checking extracted protocols **for vulnerabilities**

- **Automatic extraction techniques** to extract protocol specifications
- **AuthScan**: an end-to-end framework
- Find **7** security flaws in the **real-world** implementations

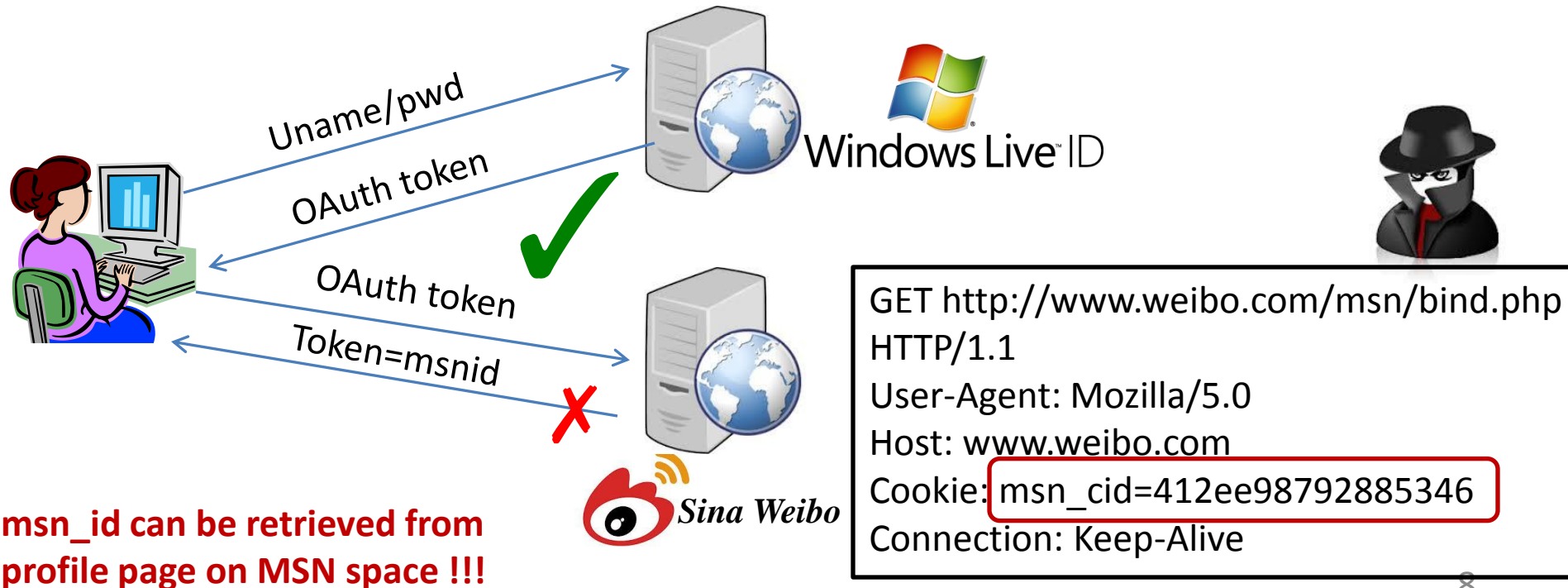
Examples #1: Freshness Problem in BrowserID Imp

- Missing Nonce
 - May lead to replay attacks



Example #2: Logic Flaw in Using Windows Live ID

- Using Publicly-Known Values as Tokens
 - Keep **constant** across multiple login sessions and the values are **publicly-known**
 - e.g., email, publicly-known id, hash(email), etc.
- Flaw found in credential cookies in Sina Weibo



Many More Vulnerability Examples

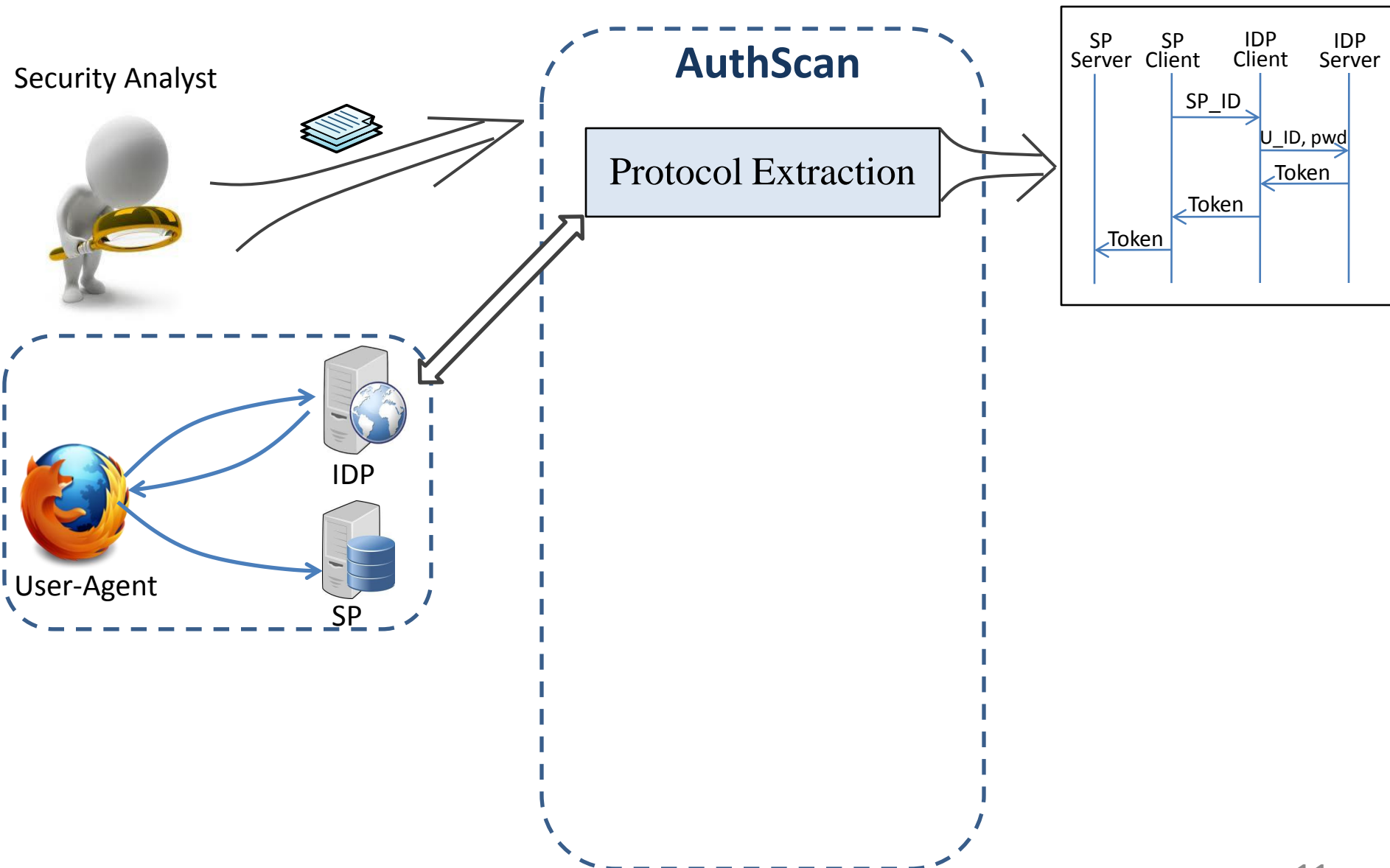
- Guessable Token
- Unchecked Referrer
 - Leading to CSRF attack
- Secret Token Leakage
- Short-length Token



Is there a **generalized** method to detect all these vulnerabilities?

Our Approach

AuthScan: Overview



Protocol Extraction & Challenge

- **Extraction:** to infer protocol from these available **code** and **messages** exchanged
 - Protocol **steps**
 - **Semantics of data element** exchanged in each step
 - Signature, cipher text, nonce, etc.
- **Challenge:** Partially available implementation
 - Partial code (client-side JavaScript code)
 - HTTP messages exchanged
- **Insight:** Hybrid Inference
 - Whitebox Program Analysis
 - Blackbox Differential Fuzzing

Whitebox Program Execution Analysis



Alice & ksad381s...nx89Ds



u, X

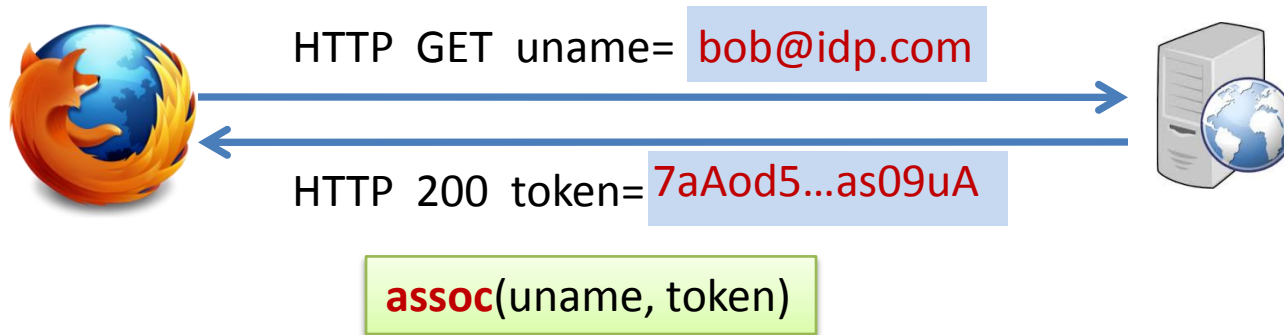
Code Example

```
window.addEventListener('message',function(event)
{
  var id=extractUser(event.data); u
  var idpSign=extractSign(event.data); X
  var data=id; u
  var idpPubKey=loadPubKey(); K
  if(verify(data, idpSign, idpPubKey)){
    {...}};
  else u X K
  {...},false);
```

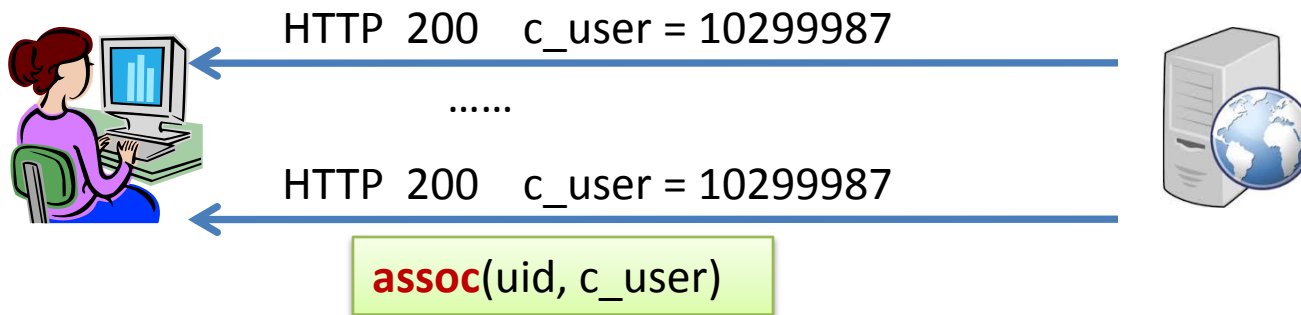
$X = \{u\}_{K^{-1}}$

Blackbox Differential Fuzzing

- To identify the relations between HTTP data

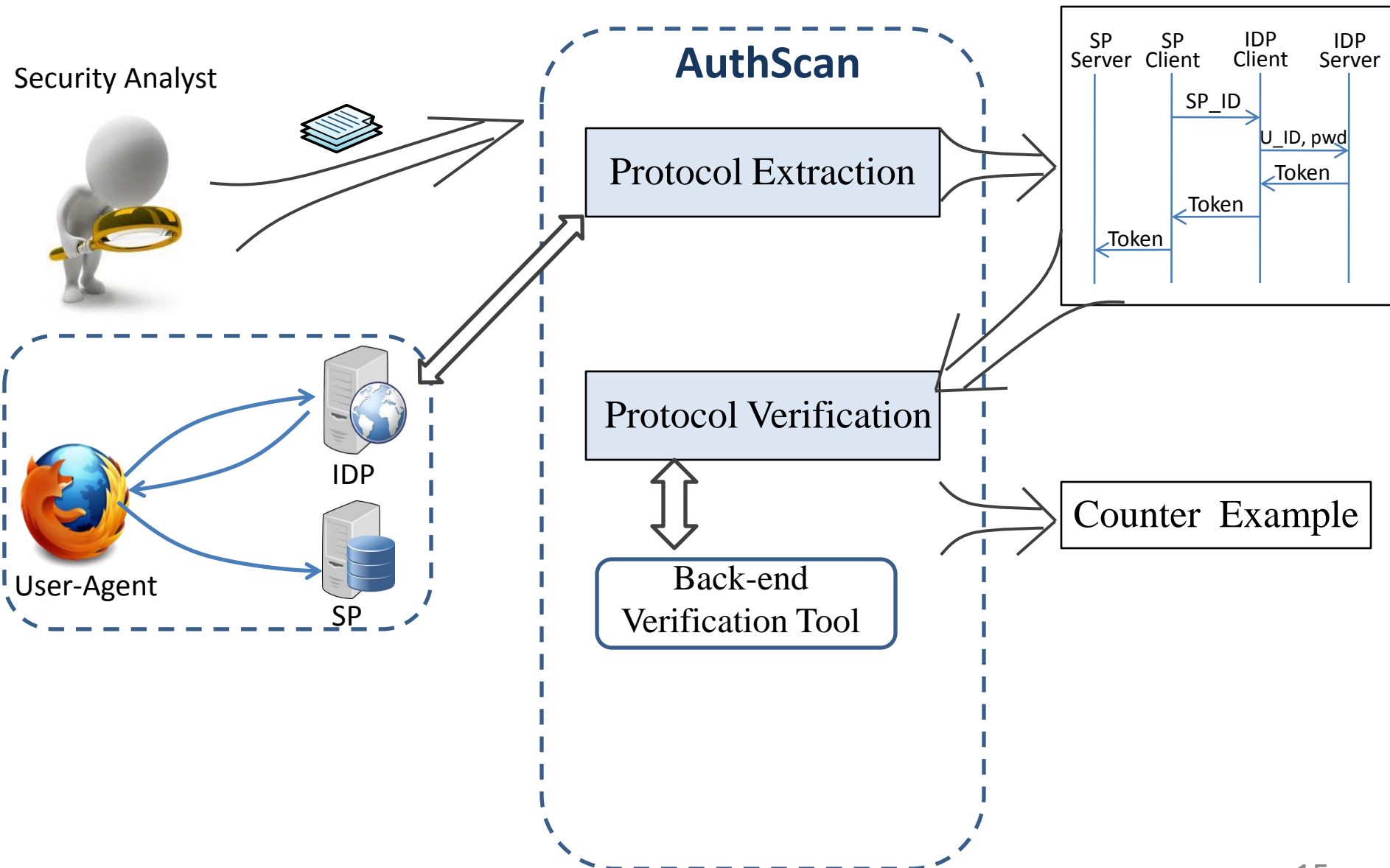


- To identify the relations between HTTP data and participants



- To eliminate the redundant messages and data
- To identify **long-lived** and **short-length** token

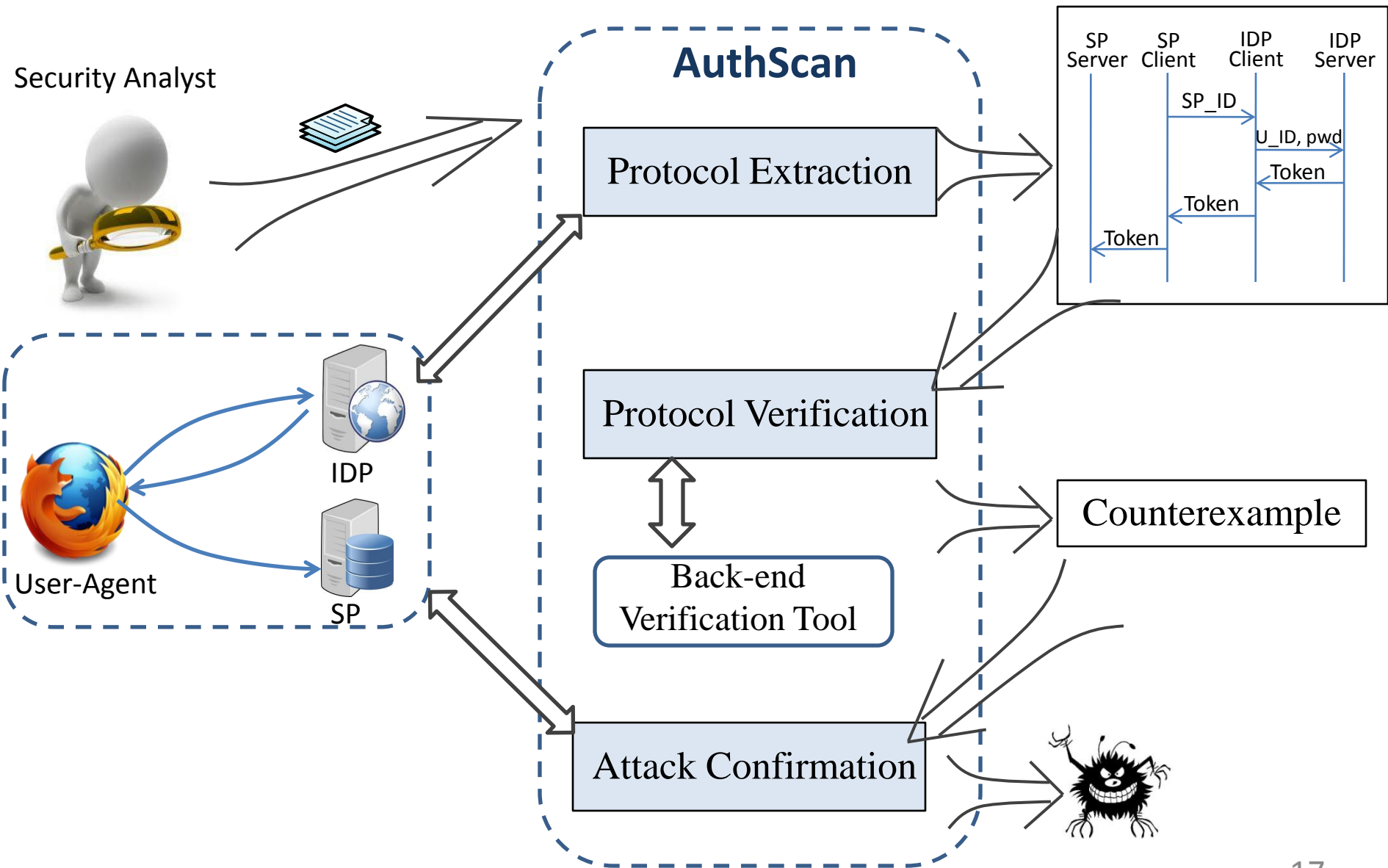
AuthScan: Overview



Attacker Models, Properties & Assumptions

- Attacker models considered in AuthScan
 - Network Attacker
 - Web Attacker
 - Same-origin policy, Referrer, postMessage
- Properties
 - Authentication
 - Correspondence [oakland' 93]
 - Secrecy
- Assumptions
 - Correct Cryptographic Algorithms
 - Knowledge of Participants
 - Each one knows the others' identifiers

AuthScan: Overview



Evaluation

AuthScan Evaluation

- Implementation
 - Implemented as a Firefox add-on
 - Uses ProVerif as the back-end
- Evaluation Subjects
 - BrowserID (three websites)
 - Facebook Connect (two websites)
 - Windows Live ID
 - Standalone websites
- Setup
 - Test harness
 - pre-registered user accounts
 - Protocol principals & public keys
 - Cryptographic functions
 - Mozilla jwcrypto used in BrowserID

Millions of Users are Impacted!

Vulnerabilities Found

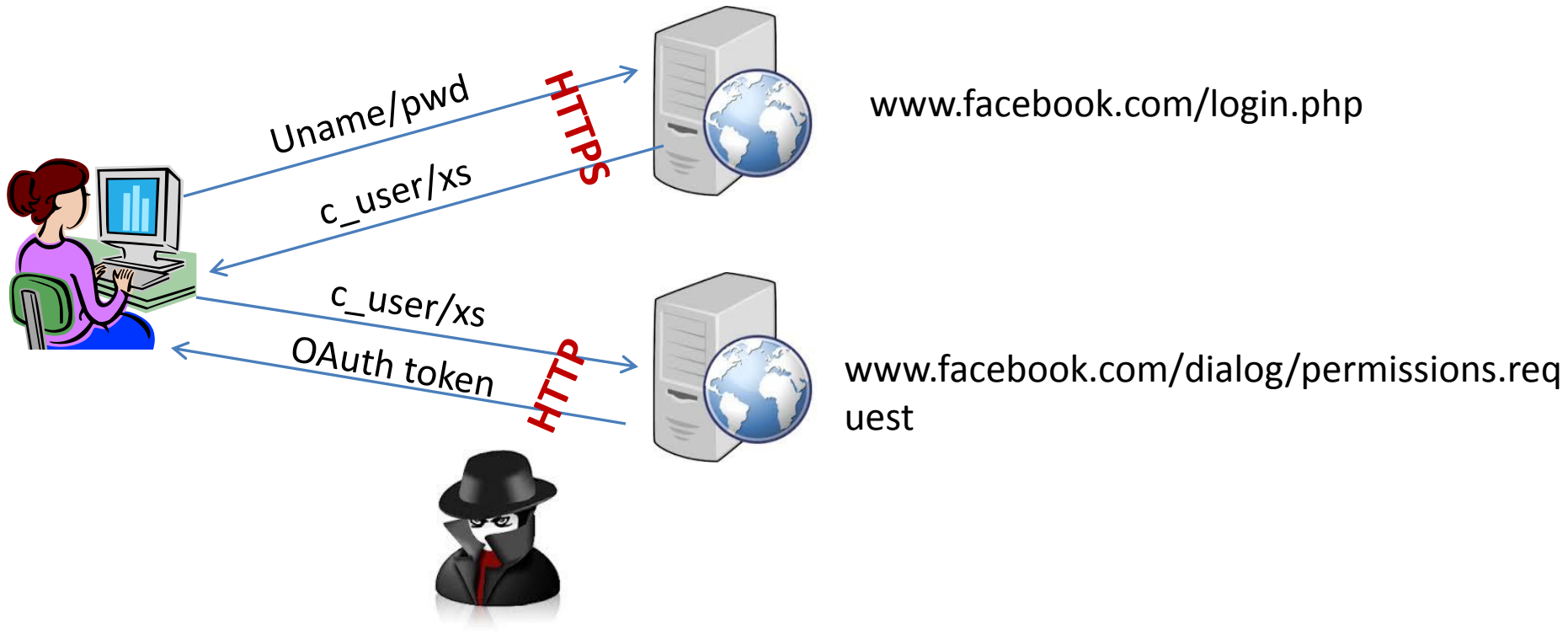
- 7 real-world vulnerabilities
 - 6 previously unknown

Web Sites	Deployed SSO	#Flaws*	Flaw Type
myfavoritebeer.com openphoto.me developer.mozilla.org	BrowserID	2(T1, T2) 2(T1, T2) 0	T1 Missing nonce in BrowserID T2 Unchecked Referrer in SPs (leading to CSRF attack)
ebayclassifieds.com familybuilder.com	Facebook Connect	2(T3, T4) 1(T3)	T3 Secret token leak in FB connect T4 Secret token leak in SP
Weibo.com	Windows Live ID	1(T5)	T5 Using Publicly-Known Values as Tokens
iyermatrimony.com	---	1(T6)	T6 Guessable Token
meetingmillionaires.com	---	1(T7)	T7 Short-Length Token

* With Overlapping

Example #3: Secret Token Leakage in FB Connect

- Secret Token Leakage
 - Secret tokens are transmitted through **unencrypted** channels
- Flaw found in secret cookie in Facebook Connect



Example #4: Guessable Token

- Guessable Token

http://www.iyertrimony.com/login/intermediatelogin.php?

sds=QdR.j/ZJEX./A&

sdss=Tf/GpQpvtzuEs&

sde=U1ZsU01UZ3dOVE01

} Keep constant

First 14 characters: keep constant

Incremented by one across accounts whose IDs are consecutive

Example #5: Short-Length Token

- Short-Length Token

http://app.icontact.com/icp/mmail-mprofile.pl?

r=36958596&l=2601&m=318326&c=752641&s=21DS



User ID

Constant among
different users' sessions

Alpha-numeric string

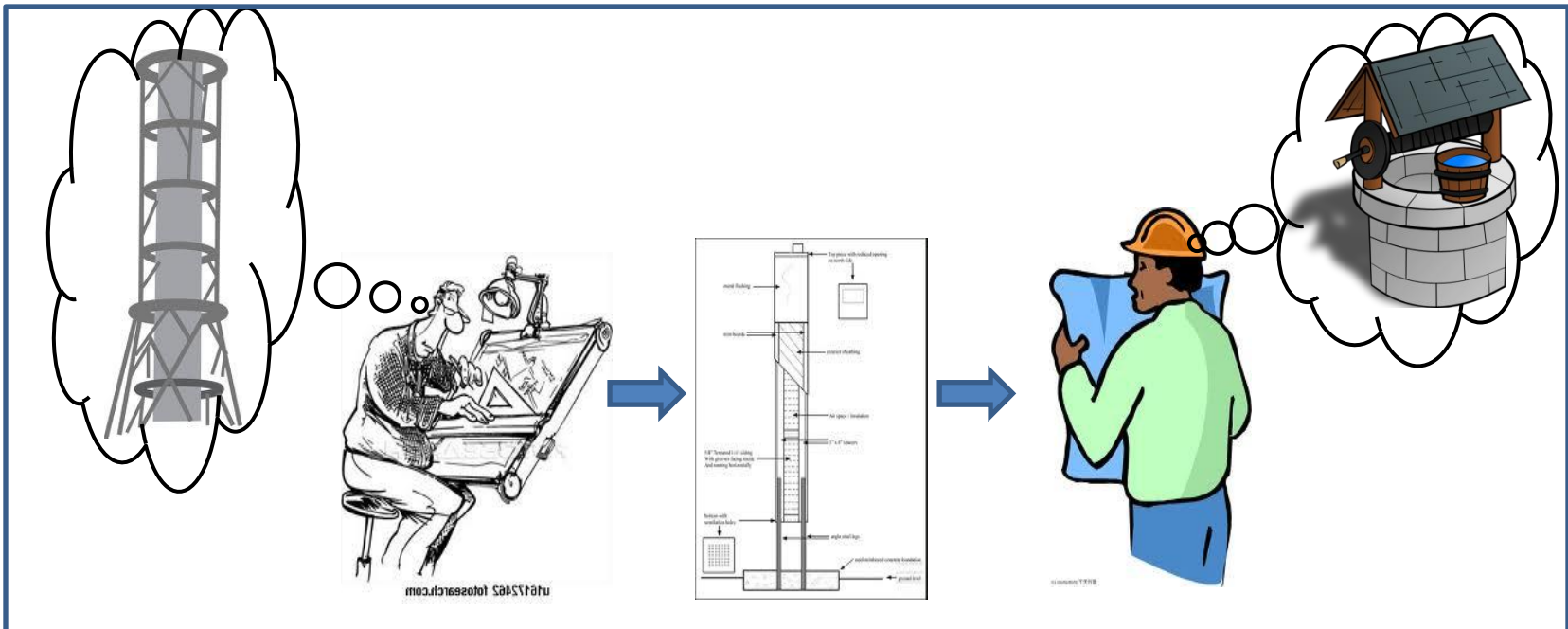
$(10 + 26)^4$ Possible Values
Attacker: 500 "probes" / min

Scalability

Web Sites	Time(s) (Excluding Verification Time)	Verification Time	Fuzzing Round
myfavoritebeer.com	113	3.0	20
openphoto.me	72	3.0	22
developer.mozilla.org	96	3.0	28
ebayclassifieds.com	127	58.7	107
familybuilder.com	110	58.7	77
Weibo.com	30	0.03	78
iyermatrimony.com	5.33	0.04	51
meetingmillionaires.com	4.72	0.04	30

Conclusion & Take-away

- AuthScan: an end-to-end framework to extract web authentication protocols from their implementations
 - Hybrid inference techniques for protocol extraction
 - Found 7 vulnerabilities in real-world web-sites
- The devil is in the details!



Reference

- **[Oakland'12]** R. Wang, S. Chen, and X. Wang.
Signing Me onto Your Accounts through Facebook and Google: a Traffic-Guided Security Study of Commercially Deployed Single-Sign-On Web Services.
- **[CCS' 12]** S.T. Sun, K. Beznosov.
The Devil is in the (Implementation) Details: An Empirical Analysis of OAuth SSO Systems.
- **[Usenix Security' 12]** S. Juraj, M. Andreas, S. Jorg, K. Marco, and J. Meiko.
On Breaking SAML: Be Whoever You Want to Be.
- **[BlackHat' 07]** E. Tsyркlevich and V. Tsyркlevich.
Single Sign-On for the Internet: A Security Story
- **[CSNT'11]** S. Pai, Y. Sharma, S. Kumar, R. M. Pai, and S. Singh.
Formal verification of OAuth 2.0 using Alloy framework.
- **[SOFSEM'11]** M. Miculan, C. Urban
Formal analysis of Facebook Connect single sign-on authentication protocol.
- **[Oakland' 93]** T. Y. C. Woo and S. S. Lam.
A Semantic Model for Authentication Protocols.

Thank you!

We are hiring!!

Phd & Post-doc in NUS, NTU and SUTD!

Contact: yangliu@ntu.edu.sg