Bringing Verification-Aware Languages and Federated Authentication to Enable Secure Computing for Scientific Communities

ILLINOIS
NCSA | National Center for Supercomputing Applications

CSL

ILLINOIS iSchool

FABRIC

TRUSTED CI
THE NSF CYBERSECURITY CENTER OF EXCELLENCE

Challenges:

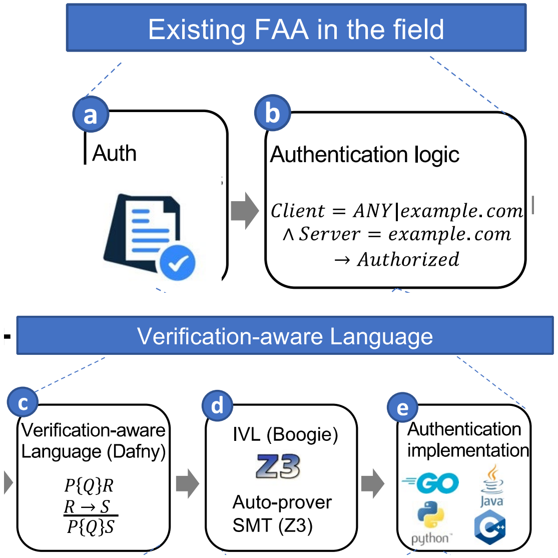
Informal natural language specifications

Complicated federated authentication logic

Wide developer experience spectrum w.r.t security knowledge

Operational, albeit unclear memory safety model in existing implementations

Significant effort in validating downstream implementations upon specs revision



Existing FAA in the field

a
Auth

b
Authentication logic

$Client = ANY | example.com$
$\land Server = example.com$
$\rightarrow Authorized$

Verification-aware Language

c
Verification-aware Language (Dafny)

$$\frac{P\{Q\}R \quad R \rightarrow S}{P\{Q\}S}$$

d
IVL (Boogie)
Z3
Auto-prover SMT (Z3)

e
Authentication implementation

Scientific Impact

Identified novel threat, failures, and uncertainty of security alerts affecting federated authentication infrastructure [IEEE QCE'24, IEEE DependSys'20, USENIX Security'24]

Formalized a subset of critical authentication functions, using Dafny, in token-based authentication based on SciTokens and FABRIC Testbed Credential Manager [Github]

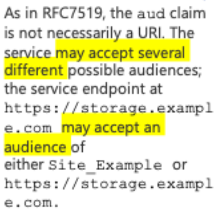
Automated program synthesis of such implementation in Python, Java, and Rust [Github]

Security testbed for reproducing authentication-related attacks [Secure-HPC Workshop @ Supercomputing'24]

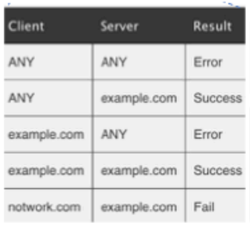
Technical Approach Demonstration

Input:

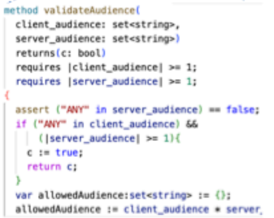
- *SciTokens specs*
- *Source repositories*
  SciTokens Python
  FABRIC credential manager
- *Knowledge of experts in SciTokens and FABRIC*



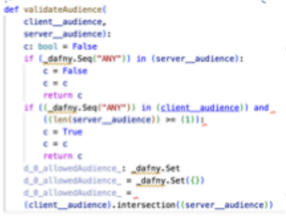
As in RFC7519, the `aud` claim is not necessarily a URI. The service may accept several different possible audiences; the service endpoint at `https://storage.example.com` may accept an audience of either `Site_Example` or `https://storage.example.com`.



Client	Server	Result
ANY	ANY	Error
ANY	example.com	Success
example.com	ANY	Error
example.com	example.com	Success
notwork.com	example.com	Fail



```
method validateAudience(
  client_audience: set<string>,
  server_audience: set<string>)
  returns(c: bool)
  requires |client_audience| >= 1;
  requires |server_audience| >= 1;
{
  assert ("ANY" in server_audience) == false;
  if ("ANY" in client_audience) &&
      (|server_audience| >= 1){
    c := true;
    return c;
  }
  var allowedAudience:set<string> := {};
  allowedAudience := client_audience * server_
```



```
def validateAudience(
    client__audience,
    server__audience):
    c: bool = False
    if (_dafny.Seq("ANY")) in (server__audience):
        c = False
        c = c
        return c
    if ((_dafny.Seq("ANY")) in (client__audience)) and
        ((len(server__audience)) >= (1)):
        c = True
        c = c
        return c
    d_0_allowedAudience_: _dafny.Set
    d_0_allowedAudience_ = _dafny.Set({})
    d_0_allowedAudience_ =
    (client__audience).intersection((server__audience))
```

Feedback to Federated Authentication & Authorization (FAA) development team

Report UNSAT (bugs)

Correct implementation

Output

- *Verified correct implementation of scoping, caching functionalities in SciTokens Python*
- *Synthesized implementations in Python, Rust, and Java*

NSF Award #2319190