CS 240
Computer Systems

#24: Tokens and SAML2 Authentication

April 19, 2022 · Wade Fagen-Ulmschneider

Security and Authentication

One advanced topic in cloud systems is security and authentication. Doing security correctly is **very hard** and the best practices change rapidly (what I learned 10 years ago is trash-tier security nowadays).

Token-Based ("Bearer") Authorization

One of the most fundamental pieces of cloud security is token-based authorization. You have seen this already:

Q: What is a **token**?

Assuming the token uses [a-zA-Z0-9], there are **62** possible character choices. What security against guessing the token does various token lengths provide?

Length	Combinations	Avg. Time to Find @ 1m guesses /sec
1	$62^1 = 62$	0.031 **ms**
2	$62^2 = 3{,}844$	1.9 **ms**
3		
4	$62^4 = 1{,}4776{,}336$	7.4 **seconds**
5	$62^5 = 916{,}132{,}832$	458 **seconds**
10	$62^{10} = 8.4 \times 10^{17}$	13 298 **years**
15	$62^{15} = 7.7 \times 10^{26}$	~12,182,899,300,000 **years**

The Google URL to this sheet worksheet:

```
1kZ45JmlvUiF8NAhHdRj9hEP8s2C7NoBTqTnYcomj99Q
1234567890123456789012345678901234567890123
4
          1         2         3         4
```

Total Length: ______ ⇒ Combinations: ____________________

Avg. Time to Find (at 1,000,000 guesses /sec):

Q: What happens if you leak the token?

...does that make token-based authentication insecure?

Token Storage

Nearly all datastores have optimizations around storing unique values, referred to as indexes in the database:

SQL Database: (Relational Datastore)	`CREATE INDEX UserToken ON tableUserTokens (token);`
MondoDB: (NoSQL Datastore)	`db.userTokens.createIndex( { "token": 1 }, { unique: true })`
Redis (Memory Datastore)	*(Every key acts like an index.)*

Tokens are stored in a BTree or HashTable-like structure, resulting in runtimes that are:

Authorization vs. Authentication
Tokens provide a form of authorization (access) to a specific resource, and are often used after a form of authentication (verification) is done.

Authentication as a Service
Many applications now rely on “Authentication as a Service” where the authentication is handled by a separate application.

- Ex: “Login with Google” / “Login with Instagram” / ...
- Ex: Queue@Illinois ⇒ Login w/ Illinois
    - Shibboleth (UIUC login technology) provides user authentication without revealing any details except that the user!

Advantages:

Disadvantages:

Almost all “Single Sign On” technologies are enabled using **Security Assertion Markup Language 2.0 (SAML2)** protocols. There are three primary “actors” in this protocol:

1. [User Agent -- **UA**]:
2. [Service Provider -- **SP**]:
3. [Identity Provider -- **IdP**]:
4. [User Artifacts]:

Service Provider (Ex: Queue@Illinois)		**User Agent** (You on Firefox)		**Identify Provider** (Univ. of Illinois)
Step 1:				
				▨
Step 2:				
Step 3:				
		▨		
Step 4:				
▨				
Step 5:				
Step 6:				
		▨		
Step 7:				
				▨

Q: When logging in with SAML2, what information is shared **directly by the user** with the service provider?

Q: What information is **shared by the identity provider** with the service provider?

Q: If your login uses 2FA, who is responsible for the 2FA?

Q: When does the service provider communicate with the identity provider directly, without the user?