# CS 340 #22: Tokens and SAML2 Authentication

Computer Systems Nov. 7, 2023 · G Carl Evans

### 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 (*much of 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: For example, the Google Doc for the next JKLMNOP is:

18N-6sQksF-RTOsvvMuvMAo_Z6veG-OdVG_OT8XzoPic						
12345678901234567890123456789012345678901234						
	1	2	3	4		

Total Length: \_\_\_\_\_  $\Rightarrow$  Combinations: \_\_\_\_\_

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

**Q:** What happens if you leak the token?

Q: What is a token?

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

Assuming the token uses [a-zA-ZO-9], there are **62** possible character choices. To optimize storage, it's common to add two more characters to create a 6-bit number (64 bits, for 64 possible characters): **base64** adds **+** and **/**, **base64url** adds **-** and **\_** instead. What security against guessing the token does various token lengths provide?

Length	Combinations	Average Time to Guess @ 1m guesses /sec
1	$64^{1} = 64$	0.032 <b>ms</b>
2	64 <sup>2</sup> = 4,096	~2 ms
3		
4	64 <sup>4</sup> = 16,777,216	~8 seconds
5	64 <sup>5</sup> = 1,073,741,824	~9 minutes
10	$64^{10} = 1.15 \times 10^{18}$	~18,000 <b>years</b>
15	$64^{15} = 1.24 \times 10^{27}$	~20,000,000,000,000 years Age of Earth: ~4,500,000,000 years

**Token Storage** 

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

SQL Database:	CREATE <mark>INDEX</mark> UserToken			
(Relational Datastore)	ON tableUserTokens (token);			
MondoDB:	db.userTokens.createIndex(			
(NoSQL Datastore)	{ "token": 1 }, { <mark>unique: true</mark> })			
<b>Redis</b> (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  $\Rightarrow$  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)		<b>User Agent</b> (You on Chrome/Firefox/)		<b>Identify Provider</b> (Univ. of Illinois)		
Step 1:						
Step 2:						
Step 3:	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?