Security II

CS 475, Spring 2018
Concurrent & Distributed Systems

Security isn't (always) free

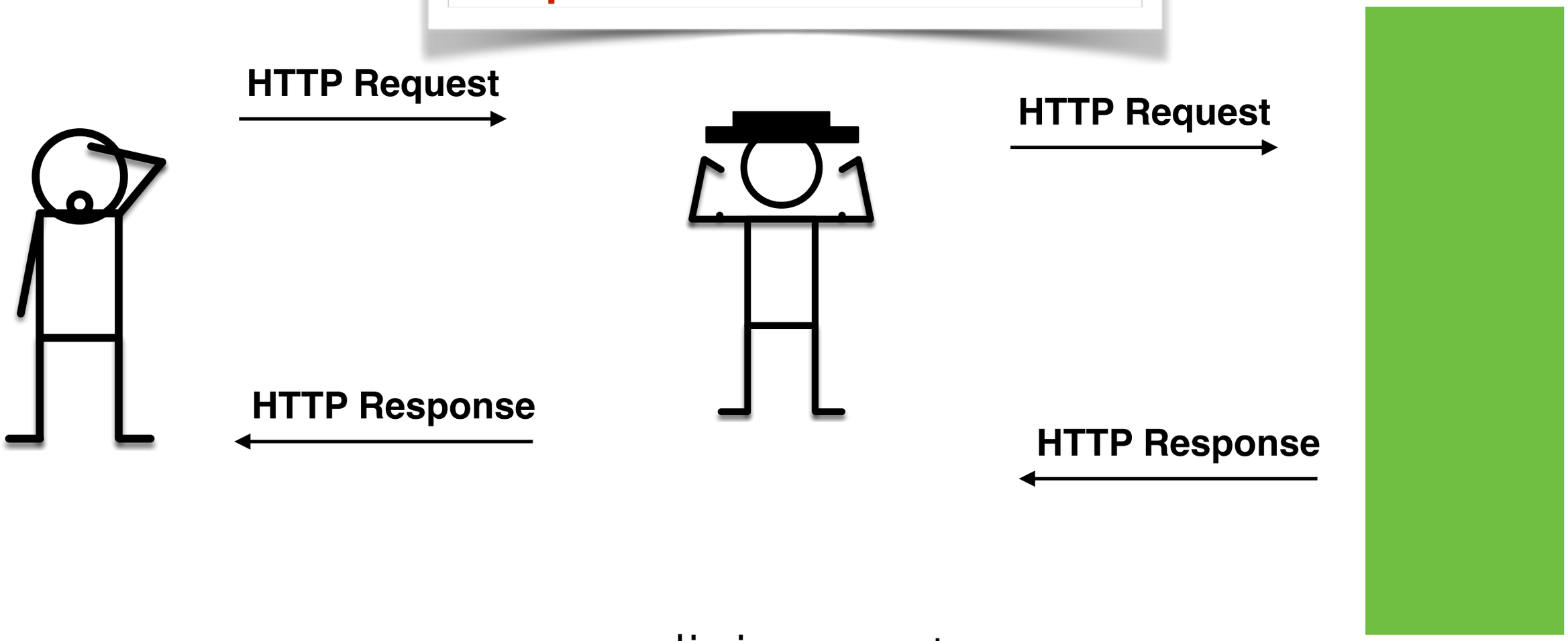
- You just moved to a new house, someone just moved out of it. What do you do to protect your belongings/property?
- Do you change the locks?
- Do you buy security cameras?
- Do you hire a security guard?
- Do you even bother locking the door?

Security: Managing Risk

- Security architecture is a set of mechanisms and policies that we build into our system to mitigate risks from threats
- Threat: potential event that could compromise a security requirement
- Attack: realization of a threat
- Vulnerability: a characteristic or flaw in system design or implementation, or in the security procedures, that, if exploited, could result in a security compromise

Example Threat: Web Server

Might be "man in the middle" that intercepts requests and impersonates user or server.



client page (the "user")

malicious actor "black hat"

server

Do I trust that this response *really* came from the server?

Do I trust that this request *really* came from the user?

Symmetric vs Asymmetric Crypto

	Symmetric Crypto	Asymmetric Crypto
Requires a pre-shared secret	Yes	No
Relative speed	Very fast	Very slow

Certificate Authorities

Amazon

amazon.com
private key

amazon.com
public key

Some [illegible] orld proof th[illegible]ve are r[illegible] amazon.com

amazon.com certificate
(AZ's public key + CA's sig)

amazon.com certificate
(AZ's public key + CA's sig)

Certificate Authority

CA private key

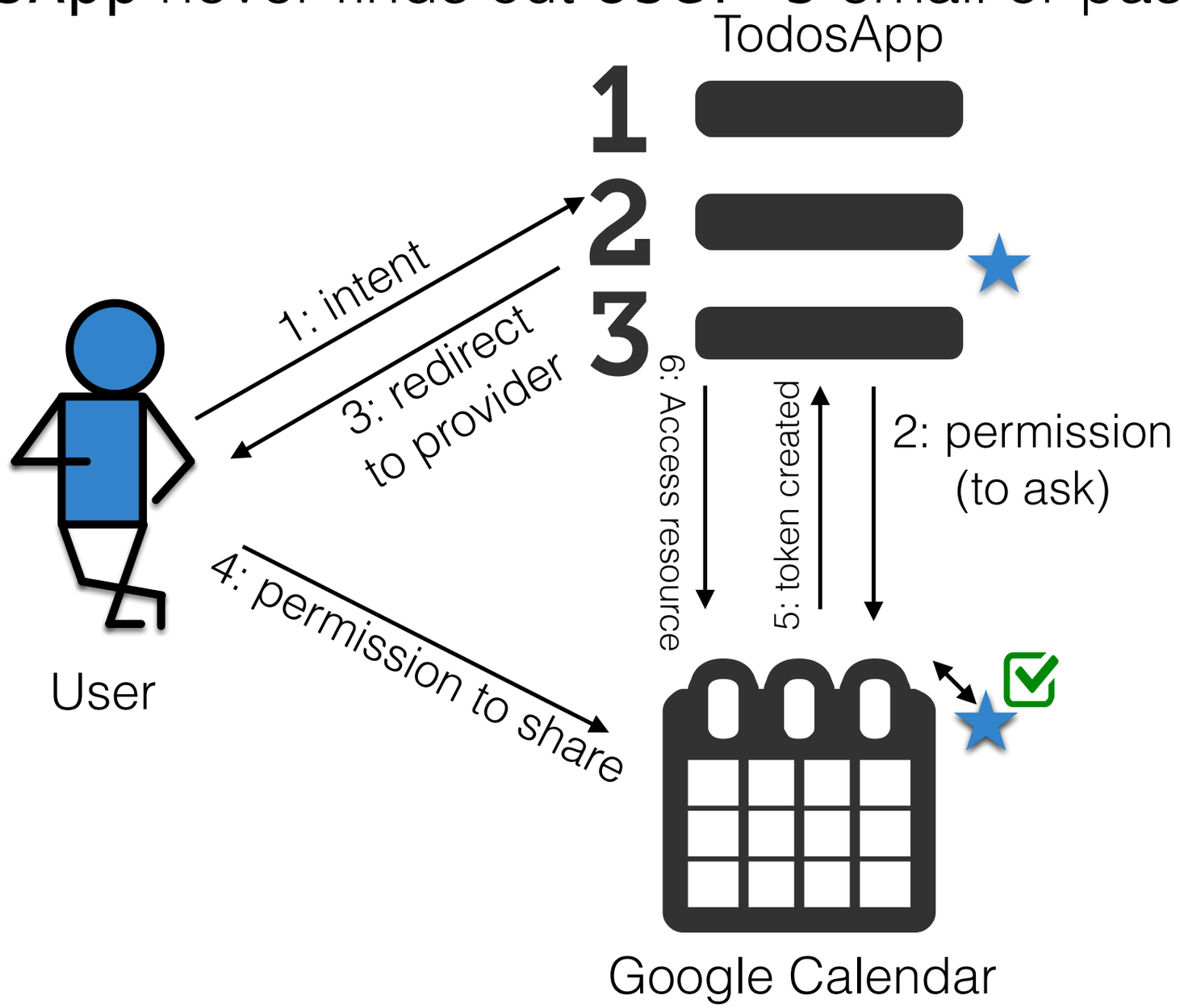
CA public key

My Laptop

CA public key

An OAuth Conversation

Goal: `TodosApp` can post events to `User's` calendar.
`TodosApp` never finds out `User's` email or password



Socrative

Class: CS475

Use your @gmu.edu email or your full name as your ID

Announcements

- Form a team and get started on the project!
  - http://jonbell.net/gmu-cs-475-spring-2018/final-project/
  - AutoLab available
- Today - more security:
  - Password schemes
  - Access control
  - DoS
  - Some slides ACK to Steve Bellovin, licensed CC BY/NC

Passwords

- How we authenticate *users* is going to vary based on our environment
- Authenticating you when you log in to your local computer is going to be different than in a distributed system, right?
- Plus: what can we use besides passwords?
  - Biometics?
  - Tokens?

Biometrics

- Advantages:
  - You can’t forget your fingers
  - You can’t lend your eyes to a friend
  - You can’t fake a fingerprint
- Why aren’t they used more?
  - Maybe they’re not that secure. . .

Biometrics

- Disadvantages:
  - False accept rate
  - False reject rate
  - Fake (or “detached”) body parts
  - Computer-synthesized voices
  - “Bit replay” (emulating the reader)
  - Non-reproducibility (matches a pattern, doesn’t create a token)
  - Many biometrics are public

Biometrics

- Biometrics work best in public places or under observation
- Remote verification is difficult, because verifier doesn't know if it's really a biometric or a bit stream replay
- Local verification is often problematic, because of the difficulty of passing the match template around
- Users don't want to rely on remote databases, because of the risk of compromise and the difficulty of changing one's body
- Best solution: use a biometric to unlock a local tamper-resistant token or chip; store keys there
  - This is what the iPhone does

Authentication Examples

- Parties: Prover (P), Verifier (V), Issuer (I)
- Issuer supplies credentials; Prover tries to log in to Verifier
- How many verifiers?
- How many different provers?
- What sort of networking is available?
- What sort of computer is P using?
- What is the relationship of P, V, and I?
- What are the adversary's powers?

Passwords: Large Enterprise

- Comparatively homegenous computing environment
- P trusts his/her own computer
- Centralized I, many Vs
- Perhaps use some pre-shared-key based system
  - Uses password as cryptographic key
  - Uses centralized database of plaintext keys (but not passwords)
  - Little risk of keystroke loggers
  - Use management chain to authorize password recovery

Passwords: Wireless ISP

- Unsophisticated user base
  - Low cost is very important
  - Trusted, high-speed internal network
- Separate login and email passwords
- Store the wireless login password on the user's machine; maybe email password, too—must avoid help-desk calls
  - Use password hints; maybe even let customer care see part of the password or hints
- Reasonably low risk of password file compromise: file theft may be less of a risk than keystroke loggers
- Many Vs for login; several Vs for email. Use centralized back-end database, with no crypto

Passwords: University Computing

- Central V database
- Wireless networking
- Very heterogenous client computers
  - Pre-shared-keys not usable; too many different client machines
  - Serious danger of eavesdropping; use encrypted logins only
  - Use back-end process to distribute password database, or use online query of it
- Classical password file may be right

Passwords: Consumer Website

- Low-value logins
- Can’t afford customer care
- Use email addresses as login names; email new password on request (but why not send out old password?)
- Don’t worry much about compromise

Passwords: Mailing list server

- Use of password is rare (and often non-existent)
- Solution: auto-generate passwords; email them to users in the clear
- No serious resources at risk, especially for public mailing lists
- Better choice than asking users to pick a password
  - people will reuse some standard password
- But—the password may give access to the archives for closed mailing lists

Passwords: Financial Services Site

- High-value login
- Protecting authentication data is crucial
- Customer care is moderately expensive; user convenience is important, for competitive reasons
  - Perhaps use tokens such as SecurID, but some customers don't like them
    - Today, perhaps use smart phones as second factor
    - Do not let customer care see any passwords
- Require strong authentication for password changes; perhaps use physical mail for communication
- Guard against compromised end-systems

Passwords: Military

- Captive user population—and they'll be there for a few years
  - User training possible
- High value in some situations
- Everyone has to carry ID anyway
  - Convert dog tag to smart card containing public/private key pair
  - Use it for physical ID (Geneva Convention) and for computer login
  - Use PIN to protect private key

Passwords: Military

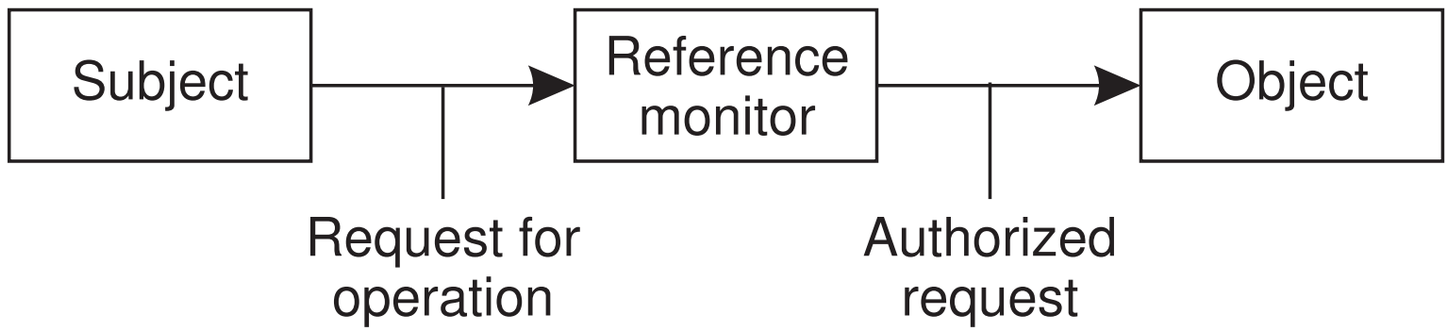
- Prisoners of war must show their dog tags
- That same device can provide access to sensitive computer systems
- POWs can be “pressured” to disclose their PINs
- Result: some pilots in Iraq in 2003 destroyed the chip before missions
- The designers forgot one thing: the risk of physical capture of the device and the device owner

Authentication - High level

- The many different forms of authentication have a great deal in common:
  - Secondary authentication
  - Dealing with server compromise
  - Credential loss
  - Susceptibility to guessing attacks
  - Administrative infrastructure
- These pieces interact
- No perfect solution… best seems to be still… passwords

Access Control

- So far, we have talked about setting up a secure channel
- Over this secure channel, client can request operations from the server
- Requests should only be carried out if the client has sufficient access rights to do that
- General model:



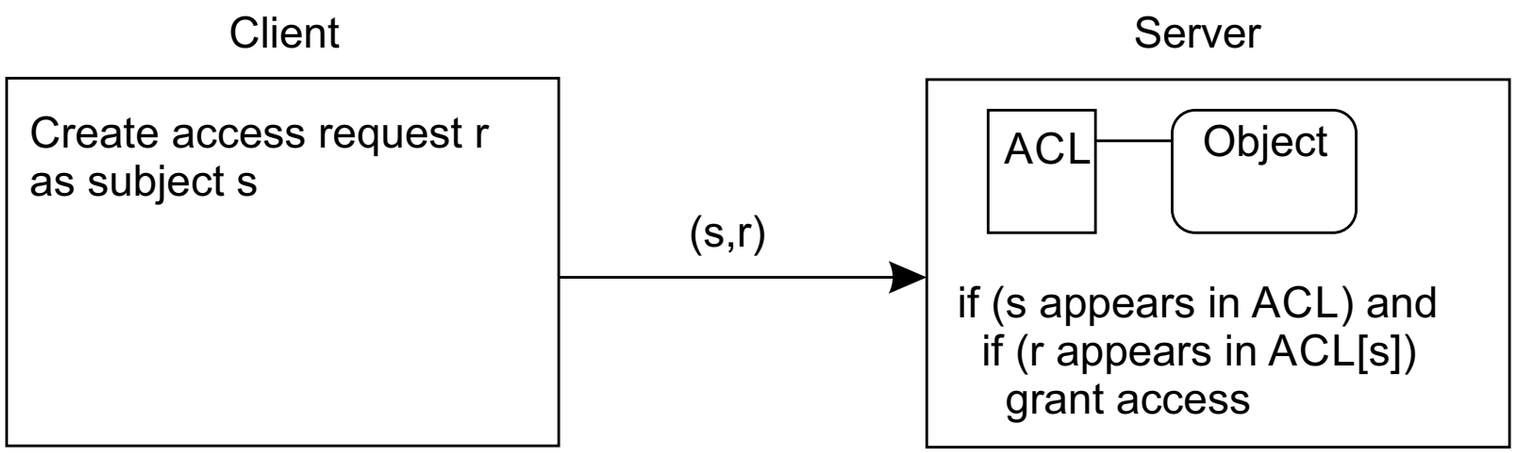
Access Control Matrix

- Models and describes the access rights of subjects to objects
- Each subject is a row, object is a column, cells list the valid operations

	File 1	File 2	File 3
Alice	rx	r	rwx
Bob	-	r	r
Charlie	rw	w	-

Access Control Lists

- In practice, nobody does this except for people modeling systems at a really high level
- Usually a very **sparse** matrix - millions of files, millions of users, users can only access their own files…
- Hence, keep a single list of permissions per object (an access control list, ACL)
- Or keep a list of capabilities per user (capability list)



Access Control Lists

	File 1	File 2	File 3
Alice	rx	r	rwx
Bob	-	r	r
Charlie	rw	w	-

Capabilitiy Lists

	File 1	File 2	File 3
Alice	rx	r	rwx
Bob	-	r	r
Charlie	rw	w	-

Access Control in Distributed Systems

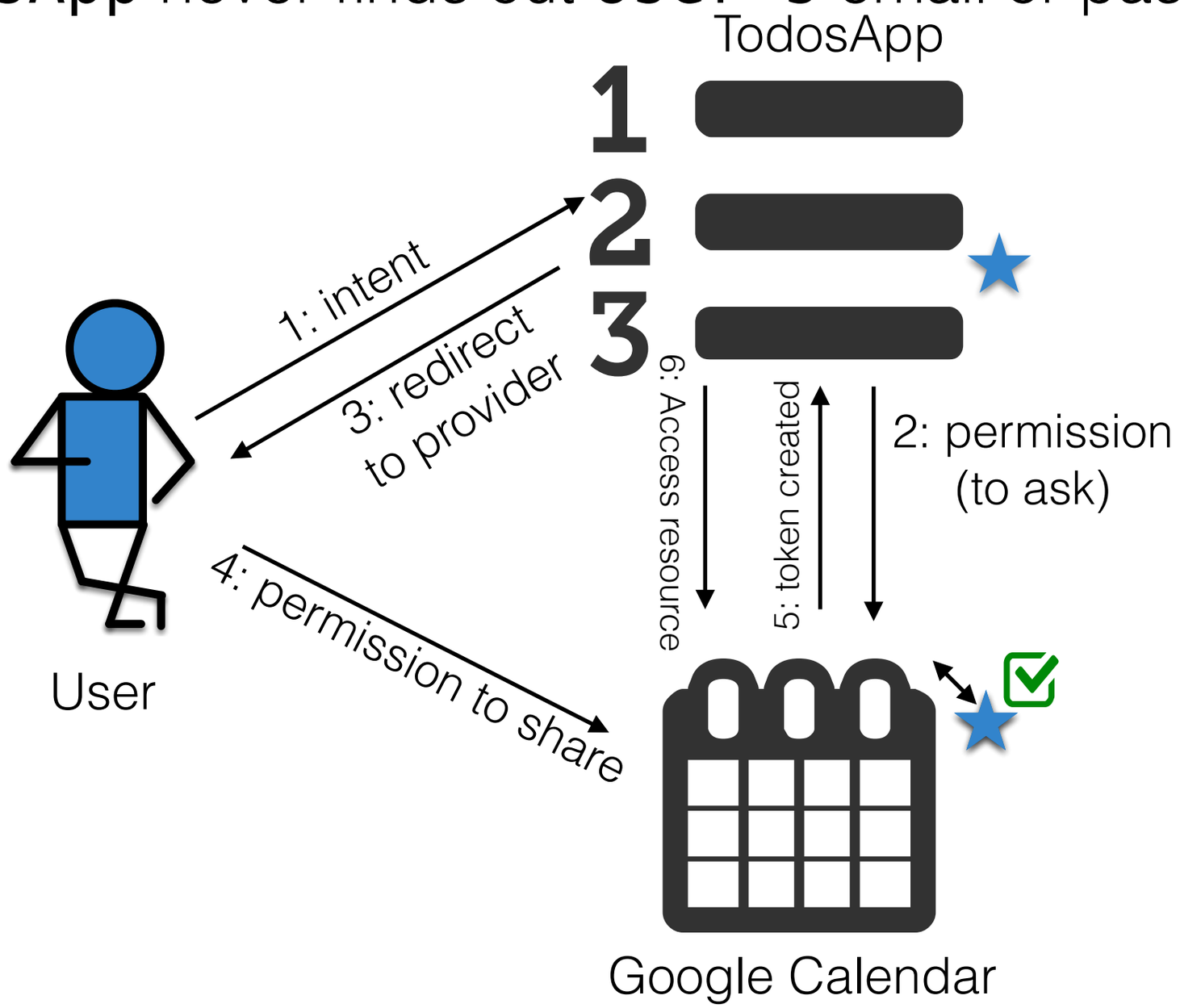
- Straightforward (?) in non-distributed systems
  - User has an account on a machine
  - That machine checks the user's access rights
- How do we do this in a distributed system?
  - Does each user have an account on each machine?
  - Single server that everyone talks to?

Delegation

- Alternative to having a single central sever: **delegation**
- An unforgeable data structure that gives a user some capability
  - E.g. a signed message

An OAuth Conversation

Goal: `TodosApp` can post events to `User's` calendar.
`TodosApp` never finds out `User's` email or password



Tokens?

A token is a **secret value**. Holding it gives us access to some privileged data. The token identifies our users and app.

Example token:


```
eyJhbGciOiJSUzI1NiIsImtpZCI6ImU3Yjg2NjFjMGUwM2Y3ZTk3NjQyNGUxZWFiMzI5OWIxNzRhNGVlNWUifQ.eyJpc3MiOiJodHRwczovL3NlY3VyZXRva
2VuLmdvb2dsZS5jb20vYXV0aGRlbW8tNzJhNDIiLCJuYW1lIjoiSm9uYXRoYW4gQmVsbCIsInBpY3R1cmUiOiJodHRwczovL2xoNS5nb29nbGV1c2VyY29ud
GVudC5jb20vLW0tT29jRlU1R0x3L0FBQUFBQUFBQUFJL0FBQUFBQUFBQUFBgL0JVV2tONkRtTVJrL3Bob3RvLmpwZyIsImF1ZCI6ImF1dGhkZW1vLTcyYTQyI
iwiYXV0aF90aW1lIjoxNDc3NTI5MzcxLCJ1c2VyX2lkIjoiSk1RclFpdTlTUlRkeDY0YlR5Z0EzeHhEY3VIMiIsInN1YiI6IkpNUXJRaXU5U1JUZHg2NGJUe
WdBM3h4RGN1SDIiLCJpYXQiOjE0Nzc1MzA4ODUsImV4cCI6MTQ3NzUzNDQ4NSwiZW1haWwiOiJqb25iZWxsd2l0aG5vaEBnbWFpbC5jb20iLCJlbWFpbF92Z
XJpZmllZCI6dHJ1ZSwiZmlyZWJhc2UiOnsiaWRlbnRpdGllcyI6eyJnb29nbGUuY29tIjpbIjEwOTA0MDM1MjU3NDMxMjE1NDIxNiJdLCJlbWFpbCI6WyJqb
25iZWxsd2l0aG5vaEBnbWFpbC5jb20iXX0sInNpZ25faW5fcHJvdmlkZXIiOiJnb29nbGUuY29tIn19.rw1pPK377hDGmSaX31uKRphKt4i79aHjceepnA8A
2MppBQnPJlCqmgSapxs-Pwmp-1Jk382VooRwc8TfL6E1UQUl65yi2aYYzSx3mWMTWtPTHTkMN4E-GNprp7hX-
pqD3PncBh1bq1dThPNyjHLp3CUlPPO_QwaAeSuG5xALhzfYkvLSINty4FguD9vLHydpVHWscBNCDHACOqSeV5MzUs6ZYMnBIitFhbkak6z5OClvxGTGMhvI8
m11hIHdWgNGnDQNNoosiifzlwMqDHiF5t3KOL-mxtcNq33TvMAc43JElxnyB4g7qV2hJIOy4MLtLxphAfCeQZA3sxGf7vDXBQ
```


Decoded:


```
{
    "iss": "https://securetoken.google.com/authdemo-72a42",
    "name": "Jonathan Bell",
    "picture": "https://lh5.googleusercontent.com/-m-OocFU5GLw/AAAAAAAAAAI/AAAAAAAAAH0/BUWkN6DmMRk/photo.jpg",
    "aud": "authdemo-72a42",
    "auth_time": 1477529371,
    "user_id": "JMQrQiu9SRTdx64bTygA3xxDcuH2",
    "sub": "JMQrQiu9SRTdx64bTygA3xxDcuH2",
    "iat": 1477530885,
    "exp": 1477534485,
    "email": "jonbellwithnoh@gmail.com",
    "email_verified": true,
    "firebase": {
    "identities": {
        "google.com": ["109040352574312154216"],
            "email": ["jonbellwithnoh@gmail.com"]
    },
    "sign_in_provider": "google.com"
},
    "uid": "JMQrQiu9SRTdx64bTygA3xxDcuH2"
}
```

Why tokens?

- Why not store username/password in the service?
- Why not store username/password on your computer?

Role-Based Access Control (RBAC)

- Permissions are granted to roles, not users
- Map users to roles
- David Wheeler: “Any software problem can be solved by adding another layer of indirection”
- Mapping can change; should be reasonably dynamic
- Example: substitute worker; replacement worker

RBAC

- RBAC is the mechanism of choice for complex situations
- Often, it isn't used where it should be, because it's more complex to set up.
- Example: giving your administrative assistant your email password
- Does this create new weaknesses?
  - New attack: corrupt the mapping mechanism between users and roles

Denial of Service Attacks

- A significant concern for distributed systems
- An attack on **availability** - attackers prevent legitimate users from accessing system
- Can attack:
  - Bandwidth
  - CPU
  - Memory
- Core problem:
  - Costs more to process a message than to send it

Distributed Denial of Service Attacks (DDoS)

- Model: Attacker has (hundreds of?) thousands of machines at disposal to attack
- Most common form of DoS today
- Exhausts network bandwidth
- Typically rooted in a botnet - some command and control infrastructure setup by an attacker, who then controls all of these machines

Strawman Defenses

- Make a filter list of bad addresses?
- Trace down the person responsible?

Heuristic Defenses

- Overprovision
- Black-hole routing
- Filter anomalies
- Replication

Overprovisioning

- Make a DDoS-proof site by making it far bigger than it needs to be
- Provision 100x bandwidth, 100x server capacity etc. compared to what you expect
- A losing battle: an attacker can always get more bots!

Black-Hole Routing

- Limits the impact of an attack
- ISP re-routes traffic to the target site to a black hole
- Site still goes offline
- But not crashed, other sites on servers sharing network links are OK
- Most DDoS attacks are short-lived, so clears up later

Anomaly Filtering

- DDoS traffic usually has something peculiar about it…
  - Automatically generated requests following a pattern?
- Route all traffic through black-box filters that try to learn this stuff and identify anomalies
- Imperfect, but often works

Other DoS attacks

- Reflector
- Complexity

Reflector Attacks

- Exploits a publicly available service to amplify an attack
- Example: DNS
- Attacker makes a (relatively small) DNS request
- Attacker forges their own IP address with the victim's
- DNS server responds to the victim's IP address

Complexity Attacks

- Increasingly common as we find defenses for other attacks
- Idea: Can I make one request that is 100 times as hard to process as other requests?
  - Then I only need to make 1% of the requests I would have had to otherwise, in order to get the same attack!

Billion lolz

```
<?xml version="1.0"?>
<!DOCTYPE lolz [
 <!ENTITY lol "lol">
 <!ELEMENT lolz (#PCDATA)>
 <!ENTITY lol1 "&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;">
 <!ENTITY lol2 "&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;">
 <!ENTITY lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;">
 <!ENTITY lol4 "&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;">
 <!ENTITY lol5 "&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;">
 <!ENTITY lol6 "&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;">
 <!ENTITY lol7 "&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;">
 <!ENTITY lol8 "&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;">
 <!ENTITY lol9 "&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;">
]>
<lolz>&lol9;</lolz>
```

After parsing: this document contains “lol” repeated literally a billion times… ~3GB of RAM