Macaroons

What are they*, and what are they used for?

* not food

TL;DR:

A macaroon is a fancy cookie :)

What are Cookies?

- small files stored on a user's computer
- holds modest amount of data specific to a particular client and website
- allows server to deliver page tailored to a particular user

What are macaroons?

- "Cookies with Contextual Caveats for Decentralized Authorization in the Cloud"
- similar enough to cookies that most developers can use them right away
- a lot of features that make them much safer to use

Features:

- Delegation
- Attenuation
- Proof-Carrying
- Third-Party Caveats
- Simple Verification
- Decoupled Authorization Logic

Delegation:

- Like cookies, macaroons can be used to delegate access rights
- Difference: Can limit when, where, and by whom the delegated authority can be exercised
- Examples: Within one minute, from a machine that holds a certain key, or by a certain logged-in user

How it works:

User1 creates macaroon --> gives macaroon to User2 --> User2 can now act on behalf of User1

Attenuation:

- Caveats can be added to the macaroon (how, when, and where)
- restricts how the access rights may be used, making them a lot safer than cookies

Example:

Start-Up "ShadyCorporation" wants to use the cookie access rights

"Just for your address book, we swear!"

If the application supports macaroons, this becomes a lot safer, as the extent on HOW, WHEN and WHERE the rights may be used can be restricted as necessary

--> Sad data-mining start-up :)

Proof-Carrying:

- Macaroons carry their own cryptographically secured proof of authorization
- Caveats are constructed chained HMAC functions (easy to add, impossible to remove)

Third-Party Caveats:

- Specifies predicates enforced by third parties
- Macaroon is only authorized if predicate is satisfied

Simple Verification:

- No hard-coded policy
- Instead agnostic verifier separate from policy

Decoupled Authorization Logic:

- In application, separates policy from mechanism
- verifier simply observes policy (in form of embedded proof) and certifies that the proof is correct
- policy is specified when macaroon is created, attenuated, and shared

How to Macaroon

example from https://github.com/rescrv/libmacaroons (https://github.com/rescrv/libmacaroons)

Create macaroon

In [ ]:
```
>>> import macaroons
>>> secret = 'this is our super secret key; only we should know it'
>>> public = 'we used our secret key'
>>> location = 'http://mybank/'
>>> M = macaroons.create(location, secret, public)
```

public portion informs us about used secret

In [ ]:
```
>>> M.identifier
'we used our secret key
```

Location tells us where macaroon can be used

In [ ]:
```
>>> M.location
'http://mybank/'
```

Signature is used for adding caveats and verification

In [ ]:
```
>>> M.signature
'e3d9e02908526c4c0039ae15114115d97fdd68bf2ba379b342aaf0f617d0552f'
```

Adding caveat

In [ ]:
```
>>> M = M.add_first_party_caveat('account = 3735928559')
```

Share macaroon by serializing it (pure ASCII)

In [ ]:
```
>>> M.serialize(format=1)
'MDAxY2xvY2F0aW9uIGh0dHA6Ly9teWJhbmsvCjAwMjZpZGVudG...'
```

In [ ]:
```
>>> print M.inspect()
location http://mybank/
identifier we used our secret key
signature e3d9e02908526c4c0039ae15114115d97fdd68bf2ba379b342aaf0f617d055
2f
```

Signature changes with each added caveat

In [ ]:
```
>>> M = M.add_first_party_caveat('time < 2020-01-01T00:00')
>>> M.signature
'b5f06c8c8ef92f6c82c6ff282cd1f8bd1849301d09a2db634ba182536a611c49'
```

In [ ]:
```
>>> M = M.add_first_party_caveat('email = alice@example.org')
>>> M.signature
'ddf553e46083e55b8d71ab822be3d8fcf21d6bf19c40d617bb9fb438934474b6'
```

In [ ]:
```
>>> print M.inspect()
location http://mybank/
identifier we used our secret key
cid account = 3735928559
cid time < 2020-01-01T00:00
cid email = alice@example.org
signature ddf553e46083e55b8d71ab822be3d8fcf21d6bf19c40d617bb9fb438934474
b6
```

Send macaroon by first serializing it

In [ ]:
```
>>> msg = M.serialize(format=1)
>>> # send msg to the bank
```

Verification

In [ ]:
```
>>> M = macaroons.deserialize(msg)
>>> V = macaroons.Verifier()
>>> V.verify(M, secret)
'Traceback (most recent call last):'
'Unauthorized: macaroon not authorized'
```

Verifier can be informed about caveats

In [ ]:
```
>>> V.satisfy_exact('account = 3735928559')
>>> V.satisfy_exact('email = alice@example.org')
```

Use cases

Same as cookies:

- Session management
- Personalization
- Tracking

... but better!

Example:

- Data store provides macaroons, authorized if and only if the application's authentication service says that the user is authenticated
- User obtains a proof that they are authenticated from service, and presents proof alongside original macaroon to storage service
- Storage service can verify that user is authenticated, without knowing anything about authentication service's implementation
- Standard implementation: storage service can authorize request without communicating with authentication service.

Thank you for your attention :)

Any questions?